Intel® Ethernet Controller E810

Datasheet

Ethernet Products Group (EPG)

PRODUCT FEATURES

General

- Support both 50Gb/s PAM4 and 25Gb/s NRZ (E810-CAM2/CAM1 only)
- Dynamic Device Personalization (DDP) with fully-programmable pipeline that can add or modify protocols on-demand, allowing for fast-paced innovations
- Application Device Queues (ADQ) feature to increase application predictability, reduce application latency, and improve application throughput
- Support both iWARP and RoCEv2 RDMA, selectable via software per port for low latency, high throughput workload
- Support for 8x10GbE connections for appliance designs (E810-CAM2 only)
- Optimized Tx-Scheduler for steady Tx traffic flow to avoid burst send and in-network congestion
- IEEE 1588 support for precision time measurement
- Enhanced Data Plane Development Kit (DPDK) for Network Functions Virtualization acceleration, advanced packet forwarding, and highly-efficient packet processing
- More resources to support higher density server virtualization deployments: 256 VFs, 768 VSIs
- Flat NVM structure with dual banks to facilitate Any-to-Any NVM update and allow rolling back to previously known good NVM when device cannot load from current active bank
- Extensive test and validation with ecosystem devices including switches, cables and transceivers for interoperability

Three SKUs

- E810-CAM2:25x25mm; PCI Express v4.0/v3.0 x16/x8; 2x100Gb, 2x50Gb, 4x25Gb, 8x10Gb, 50Gb PAM4/25Gb NRZ SerDes
- E810-CAM1:25x25mm; PCI Express v4.0/v3.0 x16/x8; 1x100Gb, 2x50Gb, 4x25Gb, 50Gb PAM4/25Gb NRZ SerDes
- E810-XXVAM2:21x21mm; PCI Express v4.0/v3.0 x8; 1x50Gb, 2x25Gb, 25Gb NRZ SerDes

Storage Networking

- Data Center Bridging (DCB)
- Stateless L3/L4 offloads for iSCSI, NAS, NFS
- Server Message Block (SMB)

NFV and Network Virtualization Overlay (NVO) Support

- Dynamic Device Personalization (DDP) with programmable pipeline for flexible frame format support
- Intelligent Flow Direction: Receiver Side Scaling (RSS), Intel® Ethernet Flow Director, Application Device Queues (ADQ)
- Comprehensive Network Virtualization Overlay protocols support.
- vSwitch Assist
- QoS: Priority-based Flow Control (802.1Qbb); Enhanced Transmission Selection (802.1Qaz); Differentiated Services Code Point (DSCP)

Server Virtualization Support

- SR-IOV: 8 PFs, 256 VFs, 256 Queues per PF, 2K queue pairs total, 768 VSIs
- Adaptive VF driver
- Scalable IO Virtualization (768 VDEVs)
- Programmable Virtual Ethernet Bridging (VEB) with ACL
- Virtual Machine Device Queue (VMDq); Virtual Machine QoS (VMQoS)

Remote Direct Memory Access (RDMA)

- Both iWARP and RoCEv2 support selectable via software per port

Precision Clocks Synchronization

- IEEE 1588 v1 and v2 PTP/802.1AS

Management

- Firmware Management Protocol (FMP)
- MCTP over PCIe and SMBus/I^2C
- PLDM over MCTP; PLDM Monitoring; PLDM Firmware Update; PLDM for RDE
- NET2BMC/OS2BMC

Security

- Hardware-based Root of Trust
- Authentication on Read and Power On; Whitelist approach on NVM update
- Built-in detection of firmware/critical setting corruption with automated device recovery

Revision 2.1
January 2021
613875-003

Revision History

Revision	Date	Notes
2.1	January 22, 2021	Updates include the following: • Updated Table 1-1, "E810 Features Summary" under the Security category. • Added Section 1.4.12.1, "Protect, Detect, and Recover". • Updated Table 1-2, "Standards Supported by the E810" to add Security category. • Updated Section 16.3.1.1, "Power On/Off Sequence". • Updated Table 16-8, "E810-CAM2/CAM1 MAX Power - PCIe Gen 4 and PCIe Gen 3". • Updated Section 16.5.5, "Reference Clock Specification" to enhance HCSL and V_{diff} discussion. • Updated Table 18-2, "E810 Thermal Specifications".
2.0	July 23, 2020	Second public release.
1.9[1]	July 13, 2020	Initial public release.

1. There are no previous publicly-available versions of this document.

Contents

NOTE: ***This page intentionally left blank.***

Chapter 1 Introduction

1.1 Overview

This document describes the external architecture (including device operation, pin descriptions, register definitions, and so on) for the Intel® Ethernet Controller E810 (E810), a dual-port 100 Gigabit Ethernet (GbE) Network Interface device.

This document is intended as a reference for architects, logic designers, firmware and software device driver developers, board designers, test engineers, or anyone else who might need specific technical or programming information about the E810.

The E810 is Intel's first multi-speed 100 GbE controller supporting rates between 100 Mb/s and 100 Gb/s. The E810's key objective is to enable a scalable Ethernet controller suitable for Enterprise, Cloud, and Communications Service Provider applications.

The key features supported in the E810 include two 100 Gigabit Ethernet ports, 100 Gigabit throughput performance, enhanced programmable packet processing pipeline, virtualization (enhanced SR-IOV support with up to 256 VFs and backward compatibility VF driver support), new features for the communications market (fine grained scheduler, transmit head drop support, adjustment of credits according to different headers, enhanced QoS, and enhanced burst control) and RDMA (iWARP and RoCEv2).

Table 1-1 provides a summary of E810 features. For additional information on supported features, see the *Intel® Ethernet Controller E810 Feature Support Matrix*.

Table 1-1. E810 Features Summary

Category	Features
Host Interface	• Compliance with PCIe 4.0 specification. • Concurrency for 256 non-posted requests.
Network Interface	• Speeds at 2x100 GbE, 2x50 GbE, 4x25 GbE, 8x10 GbE, 8x1 GbE, and 8x100 Mb/s.
Performance	• 100 Gb/s throughput (for each Tx and Rx). • Up to 80 Mpps for Tx and 90 Mpps for Rx. • 100 Gb/s single queue performance.
Software Interface	• Base mode VF compatibility. • Tx/Rx-Queues: — 16K Tx-Queues. — 2K Rx-Queues. — Dynamic allocation of queues to functions and VSIs. • Interrupts: — 2048 interrupts vectors, allocated in a flexible manner to queues and other causes. — Multiple interrupt moderation schemes. — 20M interrupts per second. • Control Queues (also known as Admin Queues): — Mailbox Queues for PF-VF and driver-driver. — Admin Queues for software-firmware control flows. — Sideband Queues for software to access IPs inside the E810. • 256 Tx Doorbell (DB) Queues. • 512 Tx Completion Queues. • Quanta Descriptor (QD) Queue per Tx-Queue. Quanta information is also embedded in the Tx doorbell. • Programmable Rx-Descriptor fields.

Table 1-1. E810 Features Summary [continued]

Category	Features
Packet Processing	• General: — Stages of parsing, switching, ACLs, classification, and packet modification. — Programmable packet processing pipeline. — Multiple control domains. — Profile-based. — Programmable actions. — Propagation of priorities between stages. • Parser: — Parses up to 504 bytes from packet header. — Parse Graph-based. — Session-based parsing. — Programmable parse engine. • Binary Classifier (VEB Switch): — 768 switch ports (VSIs). — Programmable forwarding rules. — Storm control. • ACLs: — 8K programmable TCAM entries. — Tiling capability to n*40-bit width. • Classification Filters: — Hash-based statistical distribution. — Flow Director flow-based classification. — Flow-based identification of iWARP and RoCE flows. — Programmable rules. • Modifier: — Insertion (Tx), removal (Rx), and modification of packet VLANs. — L3 and L4 checksums and CRC.
Virtualization	• Host virtualization via VMQ and SR-IOV. • 256 SR-IOV Virtual Functions (VFs). • Stateless offloads for tunneled packets (network virtualization support). • Malicious VF protection.
RDMA	• iWARP and RoCE v2. • 256K Queue Pairs (QPs). • Send Queue Push Mode. ***Note:*** RDMA is not supported when the E810 is configured for >4-port operation. ***Note:*** Userspace Direct Access (UDA) was intended to provide userspace access queues in a general way, but this feature is not supported in the E810. UDA is available only in the kernel and is limited to iWARP connection setup and error handling. UDA is not available in userspace.
QoS	• WFQ Transmit scheduler with nine programmable layers. • Pipeline sharing and starvation avoidance. • Up to 32 Congestion Domains in the Tx and Rx paths. • QoS via 802.1p PCP or Differentiated services DSCP value. • Rx packet buffer supports at least three no-drop flow control events, shared among ports.
Communication	• Packet shaping. • Packet drops/aging + reporting. • Reduced burstiness - jitter control. • Adjustment of credits per packet based on four different header types. • Concurrent Quanta Descriptors/Legacy Scheduling.
Manageability	• SMBus operating at up to 400 Kb/s. • DMTF-compliant NC-SI 1.1 Interface at 100 Mb/s. • PLDM Types 0, 2, 5, 6 for monitoring and control, firmware update, and Redfish Device Enablement. • MCTP over PCIe and SMBus. • Enterprise-level management schemes via local BMC.

Table 1-1. E810 Features Summary [continued]

Category	Features
Power Management	• Supports PCI power management states D0, D3hot, and D3cold. • APM WoL support in D0, D3hot, and D3cold. ***Note:*** Energy Efficient Ethernet (EEE) is not currently supported in the E810. Some references might remain in this document in the event that it is enabled in the future.
Time Synchronization	• Timestamp with each Rx packet. • Selective timestamps for Tx packets. • IEEE 1588 support.
Security	• Authentication of firmware contents on load from NVM. • Implements NIST 800-193 Platform Firmware Resiliency Guidelines including Protect, Detect, and Recover.

1.2 E810 Full Chip Block Diagram

The E810 is based on a 100G Ethernet controller core block, which is used in multiple Intel products. Figure 1-1 illustrates the relationship between the controller core logic and the I/O and support functions that comprise the full E810 device.

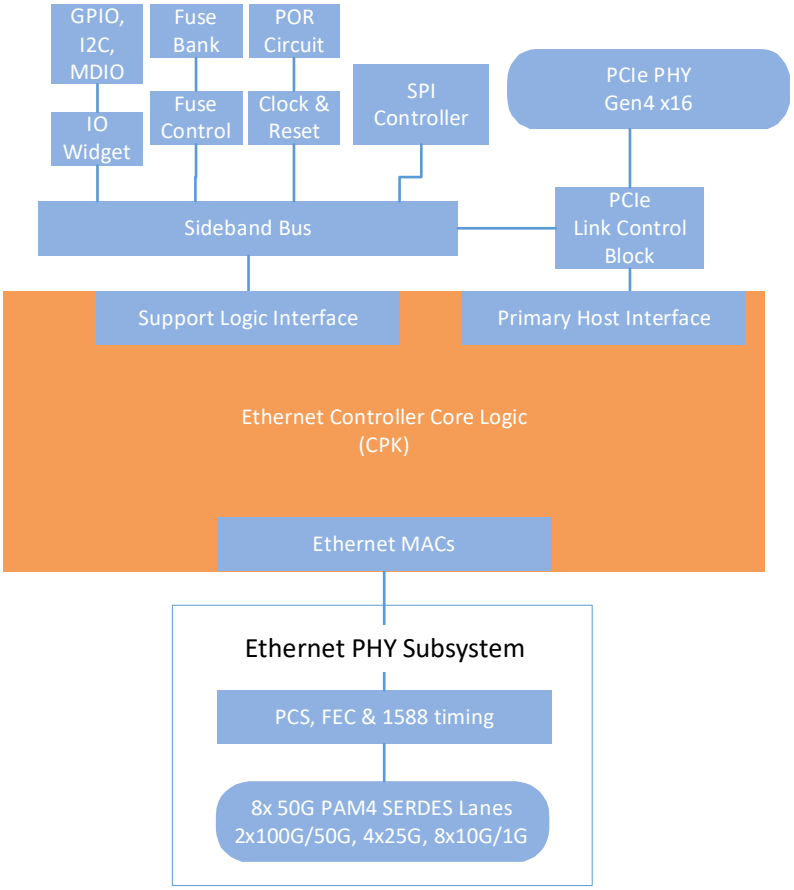


Figure 1-1. E810 Block Diagram

1.3 Controller Core Block Diagram

The controller core block diagram is depicted in Figure 1-2. In this document, references to the controller core are often abbreviated as "CPK", for example, in register bit and NVM field definitions.

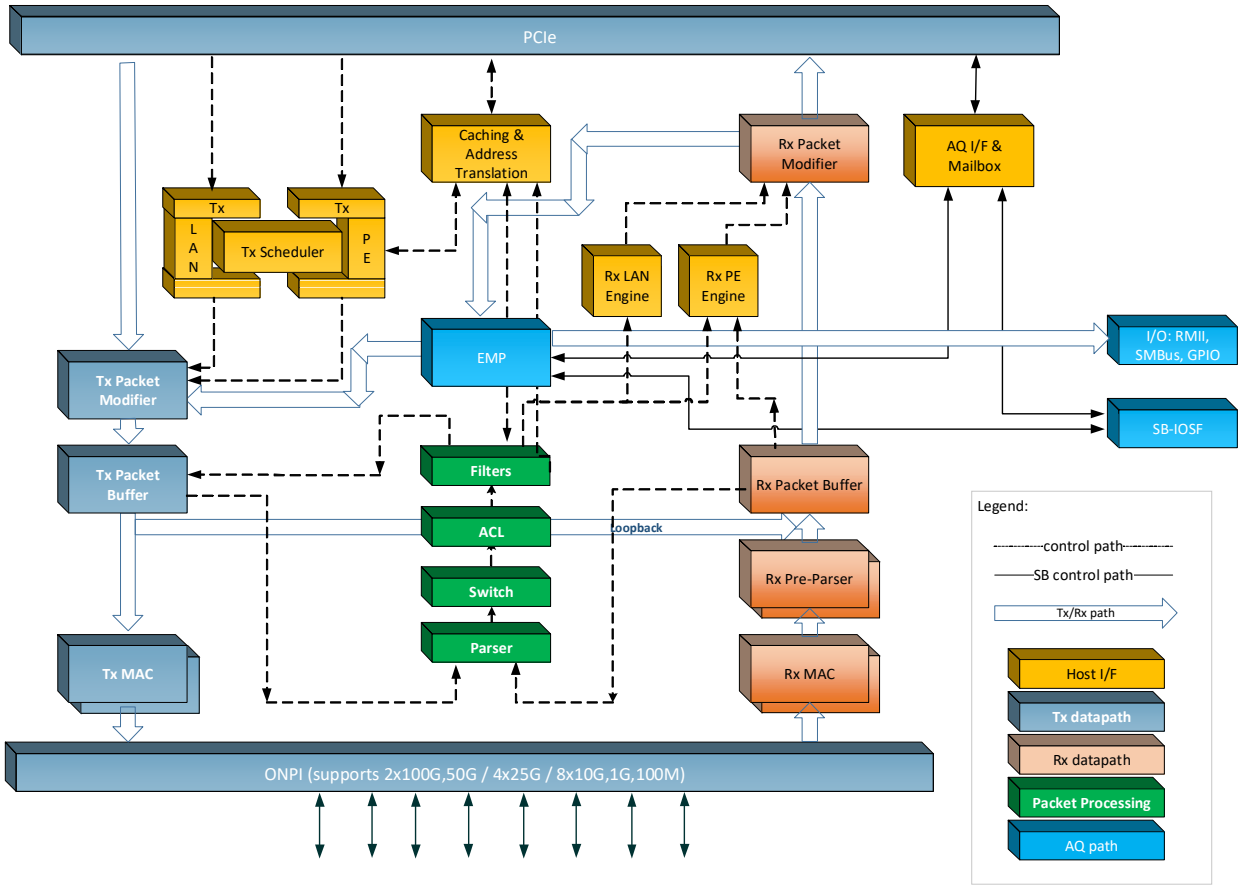


Figure 1-2. Controller Core Block Diagram

The following subsections describe the packet flows.

1.3.1 LAN Traffic Tx Flow

Transmission starts with a "doorbell" from software, indicating one packet or more is waiting in some transmit queue in host memory. If the queue is a LAN queue, the doorbell is registered with the internal Transmit Scheduler (Tx-Scheduler). If the queue is an RDMA queue, the request is registered with the Protocol Engine (PE) scheduler. When the PE scheduler selects a request, the request goes through RDMA protocol processing, at the end of which an internal request is issued to the Tx-Scheduler.

The Tx-Scheduler is responsible for pacing traffic to the network ports. It guarantees bandwidth allocation among requesters, does rate limiting where needed, and paces packets to avoid congestion later in the network.

Once the Tx-Scheduler selects a transmit queue (either from LAN or from RDMA), it services some amount of transmit data (called a "quanta"). Packet data is read from host memory and any packet processing or packet manipulation takes place. These can include partial parsing of the packet, performing Transmit Segmentation Offload (TSO), calculating relevant checksums and CRCs, L2 header manipulations UP enforcement, and inner protocol TTL decrement.

The packet is then passed to the Packet Processing Pipeline, where it queries the on-die switch to identify its destinations. A packet can be forwarded locally to a Host destination (for example VM-VM communication) or to an on-board BMC. The packet also goes through the ACL block, where permissions are checked, and might alter the forwarding decisions made in the switch.

The LAN Controller then forwards the packet to one of the LAN ports and/or to the Receive path. If the destination port is busy, the packet is stored in a local Transmit Buffer. Transmission to a LAN port is done via a 802.3 MAC that interfaces an On-Chip Network Packet Interface (ONPI).

1.3.2 LAN Traffic Rx Flow

Packet reception starts when a packet arrives to any of the ONPI channels. The packet goes through initial processing in the 802.3 MAC and a parsing/processing unit. The parsing/processing unit does some checksum verifications on the packet. The packet is then stored in the on-die Receive Packet Buffer.

The packet header then goes through the Packet Processing stage, where it potentially goes through parsing, switching, ACLs, and classification. If the packet is identified as an RDMA packet, it is passed to the Protocol Engine for protocol processing. If the packet is a LAN packet, it is forwarded to the LAN Engine. Either engine then issues a command to the Rx Modifier. The Rx Modifier fetches the packet from the Receive Packet Buffer, completes any modifications, calculates any required checksums, posts the packet into the respective Host buffer, and terminates the transaction.

1.3.3 Management Flows

Other flows include traffic to or from a system level Baseboard Management Controller (BMC). The E810 serves as a network interface to the BMC. A packet is received from the BMC via PCIe, NC-SI, or SMBus. The Embedded Management Processor (EMP) unit generates a command into the Transmit Pipeline in a similar manner to the Transmit Scheduler as described above. The Tx Modifier fetches the packet from the management interface to be processed as any Host packet.

In a similar manner, a received packet can be forwarded to the BMC once the on-die switch identifies the BMC as its destination. The packet is then forwarded to the management interface to be sent to the BMC.

Host software can also forward a packet to a system BMC. During transmit packet processing, the on-die switch identifies that the destination for the packet is the BMC interface. The E810 then forwards the packet from the transmit path to the receive path as described above. The receive path then processes the packet and forwards it to the management interface to be sent to the BMC.

1.4 Functional Blocks

This section contains a short overview The E810 functional units depicted in Figure 1-2.

1.4.1 Host Interface

The E810 implements a Gen4x16 PCIe 4.0 compliant interface.

The E810 implements a PCIe Transaction layer. The E810's PCIe host interface implements up to eight Physical Functions (PFs), and up to 256 Virtual Functions (VFs). More details on the E810's PCIe features are provided in Section 3.1, while Chapter 14 describes the PCIe programming interface.

1.4.2 Host Memory Objects

The E810 operating system drivers set up a wide variety of host memory objects that are comprehended and manipulated by the E810. All objects are set up in the context of a PCI function.

Described below are the types of memory objects:

- **LAN Transmit (Tx) and Receive (Rx) Queues** — These are ring buffers for submitting commands to the Local Area Network (LAN) engine or posting packets that arrive from the network. Commands take the form of packets/data to be transmitted, descriptors for empty host memory buffers to be filled with received packets/data, and so on. Queues are typically mapped into OS kernel space (or userspace in the case of DSI driver). In a virtualized server, they can be assigned either to the VMM, or to VMs using SR-IOV. The E810 supports up to 16K Tx-Queues and 2K Rx-Queues that can be assigned to PFs or VFs as needed. The queues assigned to a particular PCI function can be used in these important ways:
  - For distributing packet processing work to the different processors in a multi-processor system. On the transmit side, this is done by simply dedicating an independent transmit queue for each CPU to use. On the receive side, packets are classified by the E810 under operating system control into groups of conversations. Each group of conversations is assigned its own receive queue and receiving processor. Microsoft Receive Side Scaling (RSS) is one popular example of this method. ATR using the Flow Director is a another, more precise mechanism to direct Rx packets to queues.
  - For assigning Traffic Class (TC). Transmit queues assigned to different TCs are serviced at different rates by the E810 Transmit Scheduler. Receive queues assigned to different TCs can be serviced at different rates by a Quality of Service (QoS)-enabled operating system and its software device drivers.
  - To associate queues with VMs in VMDq1 or VMDq2 modes, saving packet processing and copying by the VMM
  - To associate DSI queues with different wireless operators, providing a degree of separation between the operators.
- **LAN Auxiliary structures in host memory** — These include:
  - Tx Completion Queues where write-back of transmit events can be done. The 512 Completion Queues allow coalescing of write-back information from multiple transmit queues into a small number of Completion Queues, making it easier for software to collect write-back events.

  - — Tx Doorbell (DB) Queues are used by software to coalesce Tx doorbells ("Tail Write" events) to the device, reducing load on the CPUs. Software can write a Tx DB to one of the 256 DB Queues. Once in a while, software sends an indication (also called doorbell or Tail Write) to the device, indicating that a certain DB Queue has one or more Tx-Queues' doorbells stored in it. The E810 then fetches the Tx-Queue DBs from host memory.
  - — Tx Quanta Descriptor (QD) Queues store auxiliary information on packets stored in Tx-Queues. The information assists the Transmit Scheduler by letting it know how much data is waiting for transmission in a given Tx-Queue. Each Tx-Queue has an associated QD queue.

- **Protocol Engine Queue Pairs (QPs)** — These are ring buffers (one Send Queue, one Receive Queue) for submitting commands to the Protocol Engine. Each Protocol Engine QP can be configured for either RDMA messages or for UDA packets. Commands on RDMA QPs take the form of RDMA messages to be transmitted, descriptors for empty host memory buffers to be filled with received RDMA messages, and so on. Commands on UDA QPs take the form of packets/data to be transmitted, descriptors for empty host memory buffers to be filled with received packets/data, and so on. Protocol Engine QPs are typically mapped into host userspace. In a virtualized server, they can be assigned either to the VMM, or to Virtual Machine userspace using SR-IOV. The E810 supports up to 262,144 Protocol Engine QPs, which are allocated to PFs and VFs as the E810 initializes and drivers load, and are fixed after that.
- **Control Queue (or Admin Queues (AQ)) Pairs** — AQs are used for communication between device drivers, and between a device driver and E810 embedded controllers. Each Admin Queue Pair maps one Admin Transmit Queue (ATQ) and one Admin Receive Queue (ARQ). The ATQ is a ring buffer used by the host driver for submitting commands. The ARQ conveys events to host driver that are not an immediate result of an ATQ command. The host driver posts empty buffers to the ARQ, and the E810 fills them with events. For more details on AQs, see Section 9.5.

1.4.3 LAN Engine

The LAN engine implements the host programming interface for traditional LAN traffic in both virtualized and non-virtualized scenarios. The E810 implements 768 VSIs (virtual ports) used to distribute traffic to PCI physical functions and virtual functions. These VSIs can connect to the host via 16K Tx-Queues and 2K Tx-Queues.

1.4.4 Protocol Engine

The Protocol Engine implements iWARP and RoCEv2 RDMA capability. The Protocol Engine offloads protocol processing from the host, places received data payloads directly into user buffers with no host CPU involvement, and eliminates user-kernel context switching when performing I/O by mapping the RDMA programming interface directly into application address space.

The Protocol Engine implements the latest RDMA features, including Send Queue Push Mode.

1.4.5 Transmit Scheduler

The Transmit Scheduler (Tx-Scheduler) determines the order and timing of packets sent to the network. It receives requests ("doorbells") from LAN and RDMA queues and schedules these for transmission. A scheduling decision allows a given "quanta" of bytes or packets from a given queue or set of queues to be fetched from host memory and sent to the network.

The main capabilities of the Tx-Scheduler are:

- Hierarchical scheduling tree — Scheduling decisions are done by selecting a Queue node while taking in consideration the state of its branch in each layer of the tree. Layers of the tree represent entities that define scheduling attributes, and can be a queue, a set of queues, a VSI, a PCI function, a Traffic Class, a network port, and more. Layers can also represent a network topology, scheduling to prevent congestion in nodes further down the network.
- Scheduling based on Weighted Fair Queuing (WFQ), Strict Priority (SP), and Hybrid algorithms.
- Minimum bandwidth guarantee via Dual-Rate Shapers.
- Rate limiting, including shared rate limiters.
- High-performance fine-grained and accurate scheduling.
- Fully-configurable, flexible topology, including on-the-fly configuration.
- Each scheduler node can be stopped via software-based or hardware-based flow control mechanisms.

For more detail on E810 transmit scheduler features and operation, see Section 8.3.

1.4.6 Tx and Rx Modifiers

The Tx and Rx modifiers perform operations on packet contents in the Tx and Rx pipelines, respectively. The modifiers perform updates to packet headers, such as insertion and removal of L2 tags (for example, VLAN), and calculation (and split in Rx) of header and payload checksums (insertion on Tx and validation on Rx). The Tx modifier also performs TCP Segmentation Offload (TSO).

1.4.7 Ethernet Media Access Controller (MAC)

The E810 integrates IEEE Std 802.3 compliant Ethernet MACs that operate at 100 Mb/s, 1 GbE, 10 GbE, 25 GbE, 50 GbE, and 100 GbE. A total of eight MACs map to the following speeds:

- Two MACs operating at all speeds from 100 GbE and lower.
- Additional two MACs operating at all speeds from 25 GbE and lower.
- Additional four MACs operating at all speeds from 10 GbE and lower.

Therefore, the E810 supports port configurations of 2x100 GbE, 2x50 GbE, 4x25 GbE, 8x10 GbE, 8x1 GbE, and 8x100 Mb/s.

All E810 MACs support transmission and reception of jumbo frames of up to 9728 bytes, and 802.3x flow control frames or 802.3bd priority-based flow control frames. See Section 3.2.1.5 for details.

The E810 implements a set of control signals for the PHY:

- Five independent interfaces for connection to external PHYs. Each interface can be either a Management Data Input/Output (MDIO) or Inter-integrated Circuit (I^2C) interface. These enable host software or E810 firmware to control connected external PHYs, including the ability to read and write PHY registers.
- Eight SDP signals to be used as general purpose I/O. SDPs are typically used to exchange information with or to control external devices (such as PHY devices) under E810 software driver or firmware control. See Section 3.5.1 for more details.

- Three LED indications for maximum 4-ports configuration (total of 12 pins). Each of the LED outputs can be individually configured to select which particular event, state, or activity it indicates. In addition, each LED can be individually configured for output polarity and for blinking versus non-blinking (steady-state) indications. See Section 3.5.1 for more information.

Details on how they are typically connected and used are described in Section 3.2.3.

Admin Queue commands are provided for software device driver control over the integrated PHYs.

1.4.8 Packet Parser

The E810 provides a software programmable Parser capable of supporting a wide range of well-known and proprietary protocols. The Parser examines ingress traffic, retrieves search parameters from the packet and from associated packet context, and then generates additional context based on certain packet attributes. This context is further used by other packet processing modules in the pipeline to associate the packet with a profile, a flow, an action, and so on.

The Parser supports a large set of native frame formats that are common in various networking applications. These formats are provided by the E810 out-of-the-box, not requiring any additional installation or programming. Additional frame formats can be programmed per usage.

1.4.9 VEB Switch (a.k.a. Binary Classifier)

The VEB switch identifies packet destinations for both Tx and Rx, for traffic to/from Host and to/from a manageability port (BMC). For Tx packets, it determines whether the packet goes to an external ONPI port, to a local destination either in the host or to the BMC (via loopback), or to both. For packets arriving from ONPI or from the loopback path, it selects a local destination. A local destination is named a “VSI” and is associated with a PCI function or with an EMP buffer.

The switch makes its forwarding decision based on a set of programmable rules, such as VLAN tag, a destination MAC Address, or a combination of such. Basic forwarding rules are loaded from NVM, and additional rules can be programmed per usage. Dedicated rules identify management traffic to be forwarded to a Baseband Management Controller (BMC).

The switch supports replication of broadcast and multicast packets to multiple VSIs. It also supports mirroring of both Tx and Rx packets. As any other stage in the processing pipeline, the switch generates statistics and can also insert metadata into the pipeline (potentially reaching the Rx-Descriptor).

1.4.10 Access Control Lists (ACLs)

The packet processing pipeline in the E810 provides a programmable Ternary Classifier stage for implementing functions such as Access Control List (ACL), IP Longest Prefix Match (LPM), statistics collection, and so on. The Ternary Classifier is SDN/NFV-enabled, supporting a wide range of network protocols, both standard-based and proprietary. The architecture is fully-programmable, protocol-agnostic, and action-agnostic, capable of adapting to future protocol headers and use cases.

The Ternary Classifier is located following the switch in the packet processor pipeline, as illustrated in Figure 1-2. Since the architecture of the Ternary Classifier is protocol-agnostic and action-agnostic, it can be programmed to serve as a switch extension and thus provide packet forwarding based on programmable rules, augmenting the capabilities of the switch.

1.4.11 Classification Filters

Following forwarding decision in the switch and ACLs stages, the Classification filters perform fine-grain classification of receive packets. The following capabilities are provided:

- **Hash-based classification** — Determines a destination queue in host memory through a hash function over fields in the packet headers. Several hash functions are supported, such as Microsoft RSS.
- **Flow Director** — Provides flow-level classification of packets, including association with a queue in host memory. The E810 provides on-die classification of 16K different flows.
- **Protocol Engine Quad-Hash classification** — Identifies flows to be offloaded in the on-die Protocol Engine.

The Classification Filters are programmable as are other stages in the Packet Processing Pipeline.

1.4.12 Embedded Management Processor (EMP)

The Embedded Management Processor (EMP) unit handles all management duties that cannot be performed by the E810's device drivers, and must be carried out on-chip. This includes performing parts of the E810 power-on sequence, handling AQ commands, initializing E810 Ethernet ports, participating in various fabric configuration protocols like DCBX and other Link Layer Discovery Protocol (LLDP) protocols, filling configuration requests received on one of the E810's BMC management interfaces like NC-SI, and handling special configuration requests received off an Ethernet port.

1.4.12.1 Protect, Detect, and Recover

Zero Trust is a security design strategy centered on the belief that organizations, by default, should not automatically trust anything or anybody inside or outside its perimeters and instead must verify anything and everything trying to connect and gain admittance to its systems before granting access. The Intel® Ethernet 800 Series EMP implements a design philosophy of platform resiliency with three attributes compliant with the NIST Cyber Security Framework, including NIST SP 800-193 Platform Firmware Resiliency Guidelines: Protect, Detect, and Recover.

Intel® Ethernet 800 Series uses signed firmware updates and hardware Root of Trust to protect and verify critical device settings with built-in detection of corruption, and automated device recovery to ensure the device safely returns to its originally programmed state. For details, Section 3.4.9, "NVM Authentication Procedure" and Section 15.3, "Firmware Recovery Mode".

1.4.13 Host Memory Cache (HMC)

The E810's quad hash lookup and Protocol Engine use host memory as a backing store for a variety of context objects. The Host Memory Cache (HMC) is responsible for caching and managing these context objects.

For each RDMA connection, the Protocol Engine uses a QP Context object (TCP/IP connection context that stores, for example, TCP sequence numbers) and Inbound RDMA Read Queue (IRRQ) objects that buffer inbound RDMA Read Requests until their associated Read Responses are scheduled for transmit. For each RDMA Memory Region, the Protocol Engine uses a Memory Region Table Entry (MRTE) object to store region boundary and access rights information, and a set of Physical Buffer List Entry (PBLE) objects to store virtual-to-physical address translations for this region. These and many more Protocol Engine context objects are detailed in Chapter 11.

General information on HMC operation and configuration is provided in Section 9.3.

1.4.14 Various Interfaces

1.4.14.1 Shared Serial Flash Interface

The E810 accesses the NVM via an SPI interface. The NVM device is required for storage of device firmware, device configuration parameters, identifiers that vary per adapter (like MAC Addresses), and register overrides that auto-load automatically after reset. The E810 assumes that at least 10 MB of the NVM are dedicated for these configurations.

More information on the shared serial Flash interface is available in Section 3.4.

1.4.14.2 SMBus Interface

SMBus is an optional interface for pass-through and/or configuration traffic between an external BMC and the E810. The E810's SMBus interface supports standard SMBus at 100 KHz and 400 KHz. Refer to Section 12.3.1.1 for an additional description of the SMBus interface, and Section 2.2.4 for the pin descriptions.

1.4.14.3 NC-SI Interface

NC-SI is an optional interface for pass-through and/or configuration traffic between an external BMC and the E810. Refer to Section 12.3.1.2 for an additional description of the NC-SI interface, Section 2.2.3 for the pin descriptions, and Section 12.6 for NC-SI programming.

1.4.14.4 High-Speed SDPs

On top of the eight SDPs and 12 LEDs slow interfaces accessible via the internal sideband Link, there are also four single-ended high-speed SDPs directly controlled by the E810 controller core. In addition, one differential output is directly connected to the internal clock control unit to create a high-precision clock signal. These SDPs are used for applications like IEEE 1588 that requires more precise timing of the SDPs transitions. See Section 9.7.6.1 for details.

1.5 Conventions

1.5.1 Numbers and Number Bases

Hexadecimal numbers are written with a 0x prefix (like 0xFFFF or 0x1234ABDF). Binary numbers are written with a lowercase 'b' suffix (like 1001b or 10b). Decimal numbers are indicated without any suffix or using a lowercase 'd' suffix (like 4500d).

1.5.2 Byte Ordering

This section defines the internal organization of registers and memory structures that span multiple bytes. A few conventions to start with are:

- **Network Order** — Ethernet always transmits multiple-bytes fields with the Most Significant (MS) byte first.
- **Endian Notation** — Defines how a logical entity (such as a MAC Address) is stored in a given structure (like register or descriptor). Two options exist:
  - Little Endian (LE) notation — The MS byte of the logical entity is mapped to the highest byte address of the structure.
  - Big Endian (BE) notation — The MS byte of the logical entity is mapped to the lowest byte address of the structure.

Following are some examples:

Example 1:

A 32-bit counter is equal 0x01234567 (such as the sequence number in the TCP header). The counter is transferred on the wire as: 01 23 45 67 where 01 is the first byte on the wire and 67 is the last byte.

LE registers store this counter as (in bytes) as follows:

0x01 — Highest byte address
0x23
0x45
0x67 — Lowest byte address

BE registers store this counter as (in bytes) as follows:

0x67 — Highest byte address
0x45
0x23
0x01 — Lowest byte address

Example 2:

An L2 type register that holds the value of IPv4 header is equal 0x0800. The field is transferred on the wire as: 08 00 where 08 is the first byte on the wire.

LE registers store this counter as (in bytes) as follows:

0x08 — Highest byte address
0x00 — Lowest byte address

BE registers store this counter as (in bytes) as follows:

0x00 — Highest byte address
0x08 — Lowest byte address

Example 3:

A 48-bit Ethernet MAC Address equals 0x00112348A9BE. The Ethernet MAC Address is transferred on the wire as: 00 11 23 48 A9 BE where 00 is the first byte on the wire.

LE registers store this counter as (in bytes) as follows:

0x00 — Highest byte address
0x11
0x23
0x48
0xA9
0xBE — Lowest byte address

BE registers store this counter as (in bytes) as follows:

0xBE — Highest byte address
0xA9
0x48
0x23
0x11
0x00 — Lowest byte address

The following rules determine the Endian-ness of E810 structures:

- The general rule is that all structures are defined in LE notation unless defined otherwise. These structures include:
  - Registers.
  - AQ commands.
  - Structures in host memory (including any type of descriptors).
  - NVM.
  - LAN and PE contexts.
- The following structures are in BE notation:
  - Host memory buffers that are received or transmitted.
  - Any structures that contains a MAC Address (see exception for field vector).
  - Quad hash context programming registers (GL_SWT_LOFV_PE, GL_SWT_LOFV_SW and EMP_SWT_LOFV) are each defined in LE, while register 'n' is mapped to words (63 - 2*'n') and (62 - 2 *'n') in the field vector.
  - IP Addresses in Protocol Engine host memory.
- The following structures have a mixed notation:
  - Field vector is presented in mixed BE/LE notation: Words are ordered in BE notation and bytes within the words are presented in LE notation.
  - Type-Length-Value (TLV) structures are stored in the NVM in mixed BE/LE notation: Words are ordered in BE notation and bytes within the words are presented in LE notation.

1.6 Support Documents

Table 1-2 lists industry standards relevant to the E810.

Table 1-2. Standards Supported by the E810

Category	Description
ARP	**Title:** "RFC 826: An Ethernet Address Resolution Protocol (ARP)", November 1982 **Document:** http://www.ietf.org/rfc/rfc0826.txt **Description:** Protocol to convert IP Addresses to Ethernet addresses.
Base	**Title:** "RFC 1071: Computing the Internet Checksum", September 1988 **Document:** http://www.ietf.org/rfc/rfc1071.txt **Description:** This RFC describes how to compute the Internet checksums used in IP, TCP and UDP.
Base	**Title:** "RFC 1180: A TCP/IP Tutorial", January 1991 **Document:** http://www.ietf.org/rfc/rfc1180.txt **Description:** Bare bones tutorial of TCP/IP protocol suite: ARP, IP and TCP and Upper Layer Protocols (ULPs). This is included for informative purposes only.
Base	**Title:** "RFC 1936: Implementing the Internet Checksum in Hardware", April 1996 **Document:** http: //www.ietf.org/rfc/rfc1936.txt **Description:** Techniques for efficiently implementing the Internet checksum in hardware.
DCB	**Title:** "DCB Capability Exchange Protocol Base Specification", Revision 1.01 **Document:** http://www.ieee802.org/1/files/public/docs2008/az-wadekar-dcbxcapability-exchange-discovery-protocol-1108-v1.01.pdf **Description:** Defines CEE DCBX, a pre-standard version of the DCB Capability Exchange Protocol.
DCB	**Title:** "Priority Grouping for DCB Networks (Enhanced Transmission Selection)", Revision 1.01 **Document:** http://www.ieee802.org/1/files/public/docs2008/az-wadekar-etsproposal-0608-v1.01.pdf **Description:** Defines CEE ETS, a pre-standard version of the DCB Enhanced Transmission Selection Protocol
Ethernet	**Title:** "IEEE Standard for Ethernet" (IEEE Std 802.3-2018) **Document:** Available from http://standards.ieee.org/getieee802 **Description:** Specifies the Ethernet MAC and PHY layers up to 400 Gb/s. Includes these now superseded docs (among many others): 802.3ae (10 Gb/s base spec), 802.3an (10GBASE-T), 802.3ap (backplane Ethernet, KX, KX4, KR), 802.3ba (100G), 802.3by (25G), 802.3bs (200G/400G), and more.
Ethernet	**Title:** "IEEE Standard for Ethernet – Amendment 3: Media Access Control Parameters for 50 Gb/s and Physical Layers and Management Parameters for 50 Gb/s, 100 Gb/s, and 200 Gb/s Operation" (IEEE Std 802.3cd-2018) **Document:** Available from http://standards.ieee.org/getieee802 **Description:** Defines Ethernet MAC for 50 GbE and PHY layers for 50G/100G/200G based upon 50Gb/s PAM4 electrical and optical signaling.
Ethernet	**Title:** "Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications – Amendment 8: MAC Control Frame for Priority-based Flow Control" (IEEE P802.3bd-2011) **Document:** Available from http://standards.ieee.org/getieee802 **Description:** Defines a MAC control frame to support 802.1Qbb priority-based flow control.
Ethernet	**Title:** "IEEE Standard for Local and Metropolitan Area Networks – Media Access Control (MAC) Bridges" (IEEE Std 802.1D-2004) **Document:** Available from http://standards.ieee.org/getieee802 **Description:** Base specification for Ethernet bridging.
Ethernet	**Title:** "IEEE Standards for Local and Metropolitan Area Networks - Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks" (IEEE Std 802.1Q-2011) **Document:** Available from http://standards.ieee.org/getieee802 **Description:** Ethernet VLAN-aware bridge specification.
Ethernet	**Title:** "IEEE Standard for Local and metropolitan area networks— Link Aggregation" (IEEE Std 802.1AX-2008) **Document:** Available from http://standards.ieee.org/getieee802 **Description:** Logic and protocols that enable aggregation of one or more Ethernet links into a single logical link. Until recently, link aggregation was defined in the 802.3 specification, but in the 2008 version it was moved to 802.1.

Table 1-2. Standards Supported by the E810 [continued]

Category	Description
Ethernet	**Title:** "Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks — Amendment17: Priority-based Flow Control" (IEEE P802.1Qbb-2011) **Document:** Available from http://standards.ieee.org/getieee802 **Description:** Priority-based Flow Control (PFC) is one of the specifications that comprise DCB. PFC enables flow control per TC on IEEE 802 point-to-point full duplex links. This is achieved by a mechanism similar to the IEEE 802.3 Annex 31B PAUSE, but operating on individual priorities.
Ethernet	**Title:** "Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks — Amendment 21: Edge Virtual Bridging" (IEEE P802.1Qbg-2012) **Document:** Available from http://standards.ieee.org/getieee802 **Description:** EVB is one of the specifications that comprise DCB. EVB defines many of the virtual switching features on end stations like the E810. This includes definition of things like S-channels, which enable the multiplexing of multiple virtual channels on a single physical LAN, VSIs, VEBs, VEPAs, and so on. It also defines new management infrastructure for administering the new features.
Ethernet	**Title:** "Virtual Bridged Local Area Networks — Bridge Port Extension" (IEEE P802.1BR/D3.3) **Document:** Available from http://standards.ieee.org/getieee802 **Description:** BPE is a dedicated specification describing the bridge port extender element. It specifies the use of E-channels and a multicast replication service to extend bridge ports across multiple physical or virtual devices.
Ethernet	**Title:** "IEEE Standard for Local and metropolitan area networks — Station and Media Access Control Connectivity Discovery" (IEEE Std 802.1AB-2009) **Document:** Available from http://standards.ieee.org/getieee802 **Description:** Defines Link Layer Discovery Protocol (LLDP) that enables a server to advertise its identity, capabilities, and interconnections to other entities on an Ethernet fabric.
Ethernet	**Title:** "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems" (IEEE Std 1588-2008) **Document:** Available from http://standards.ieee.org/getieee802 **Description:** Defines a protocol that enables precise synchronization of clocks in systems communicating via packet networks.
Ethernet	**Title:** "SFF-8436 Specification for QSFP+ 4X 10 Gb/s Pluggable Transceiver", Revision 4.9, 8/31/2018 **Document:** Available from https://www.snia.org/technology-communities/sff/specifications **Description:** Defines the Quad Small Form Factor Pluggable (QSFP+) module mechanicals, electrical interface and pin-out, and so on.
Ethernet	**Title:** "SFF-8431 SFP+ 10 Gb/s and Low Speed Electrical Interface", Revision 4.1, 7/6/2009" **Document:** Available from https://www.snia.org/technology-communities/sff/specifications **Description:** Defines the SERDES Framer Interface (SFI) high speed electrical interface to a Small Form-factor Pluggable (SFP+) optical module. Also defines the SFP+ management interface.
Ethernet	**Title:** "RMII™ Specification", Revision 1.2, 3/20/1998 **Document:** http://ebook.pldworld.com/_eBook/-Telecommunications,Networks-/TCPIP/RMII/rmii_rev12.pdf **Description:** Reduced pin count interface used in place of IEEE standard Media Independent Interface (MII). The E810 has an Reduced Media Independent Interface (RMII) based interface for its NC-SI connection.
Ethernet	**Title:** Serial-GMII Specification, rev 1.8, 11/2/2005, published by Cisco Systems **Document:** ftp://ftp-eng.cisco.com/smii/sgmii.pdf **Description:** Reduced pin count interface used in place of IEEE standard Gigabit Media Independent Interface (GMII).
Ethernet	**Title:** "Ethernet Alliance, Ethernet Jumbo Frames", Version 0.1, 11/12/2009 **Document:** http://ethernetalliance.org/wp-content/uploads/2011/10/EA-Ethernet-Jumbo-Frames-v0-1.pdf **Description:** Document describing jumbo frames. Included for informative purposes only.
Ethernet	**Title:** "Extended Frame Sizes for Next Generation Ethernets" **Document:** http://staff.psc.edu/mathis/MTU/AlteonExtendedFrames_W0601.pdf **Description:** Document describing jumbo frames. Included for informative purposes only.
Ethernet	**Title:** "Extended Ethernet Frame Size Support", November 1999 **Document:** https://www.ietf.org/proceedings/48/I-D/kaplan-isis-ext-eth-02.txt **Description:** Document describing jumbo frames. Included for informative purposes only.

Table 1-2. Standards Supported by the E810 [continued]

Category	Description
Ethernet (DCB)	**Title:** "Virtual Bridged Local Area Networks — Amendment 18: Enhanced Transmission Selection for Bandwidth Sharing Between Traffic Classes" (IEEE P802.1Qaz-2011) **Document:** Available from http://standards.ieee.org/getieee802 **Description:** ETS is one of the specifications that comprise DCB. ETS enables arbitration of bandwidth between TCs. This specification also defines Data Center Bridging Exchange (DCBX) protocol. DCBX enables configuration of DCB features onto an Ethernet LAN.
Ethernet/IP	**Title:** "RFC 894: A Standard for the Transmission of IP Datagrams over Ethernet Networks", April 1984 **Document:** http://www.ietf.org/rfc/rfc894.txt **Description:** This specifies the method for transmitting IP datagrams over Ethernet.
Ethernet/IP	**Title:** "RFC 1042: A Standard for the Transmission of IP Datagrams over IEEE 802 Networks", February 1988 **Document:** http://www.ietf.org/rfc/rfc1042.txt **Description:** This specifies the method for transmitting IP datagrams over IEEE 802.3 networks. Obsoletes: RFC 948
IP	**Title:** "RFC 791: Internet Protocol", September 1981 **Document:** http://www.ietf.org/rfc/rfc0791.txt **Description:** Base specification for IPv4.
IP	**Title:** "RFC 815: IP Datagram Reassembly Algorithms", July 1982 **Document:** http://www.ietf.org/rfc/rfc0815.txt
IP	**Title:** "RFC 2460: Internet Protocol, Version 6 (IPv6) Specification", December 1998 **Document:** http://www.ietf.org/rfc/rfc2460.txt **Description:** Base specification for IPv6. Obsoletes RFC 1883.
IP	**Title:** "RFC 2474: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", December 1998 **Document:** https://tools.ietf.org/html/rfc2474 **Description:** Obsoletes RFC 1349, 1455.
IP	**Title:** "RFC 2710: Multicast Listener Discovery (MLD) for IPv6", October 1999 **Document:** http://www.ietf.org/rfc/rfc2710.txt **Description:** Specifies the protocol used by an IPv6 router to discover the multicast listeners on its directly attached links. MLD is derived from version 2 of IPv4's Internet Group Management Protocol, IGMPv2. This is an important standard for power management.
IP	**Title:** "RFC 3810: Multicast Listener Discovery Version 2 (MLDv2) for IPv6", June 2004 **Document:** http://www.ietf.org/rfc/rfc3810.txt **Description:** Updates RFC 2710. This is an important standard for power management.
IP	**Title:** "RFC 2873: TCP Processing of the IPv4 Precedence Field", June 2000 **Document:** http: //www.ietf.org/rfc/rfc2873.txt **Description:** Corrects a conflict between TCP as defined in RFC 793 and DiffServ in handling of the IPv4 precedence field.
IP	**Title:** "RFC 4861: Neighbor Discovery for IP version 6 (IPv6)", September 2007 **Document:** http://www.ietf.org/rfc/rfc4861.txt **Description:** IPv6 nodes use neighbor discovery to discover each other's presence, to determine each other's link-layer addresses, to find routers, and to maintain reachability information. This is an important standard for power management. Obsoletes RFC 2461.
Mgmt	**Title:** "System Management Bus (SMBus) Specification", v3.0, December 20, 2014 **Document:** Available from http://smbus.org/specs/ **Description:** A two-wire interface for communication of management information, based on the principles of operation of I^2C.
Mgmt	**Title:** "DSP0222: Network Controller Sideband Interface (NC-SI) Specification", Version 1.1.0, September 23, 2015 **Document:** http://dmtf.org/sites/default/files/standards/documents/DSP0222_1.1.0.pdf **Description:** Standardizes the sideband communication interface between a NIC (the E810) and a BMC.

Table 1-2. Standards Supported by the E810 [continued]

Category	Description
Mgmt	**Title:** "DSP0236: Management Component Transport Protocol (MCTP) Base Specification", Version 1.3.0, November 24, 2016 **Document:** http://dmtf.org/sites/default/files/standards/documents/DSP0236_1.3.0.pdf **Description:** Specifies MCTP protocol.
Mgmt	**Title:** "DSP0237: Management Component Transport Protocol (MCTP) SMBus/I^2C Transport Binding Specification", Version 1.0.0, May 21, 2017 **Document:** http://dmtf.org/sites/default/files/standards/documents/DSP0237_1.1.0.pdf **Description:** Describes the binding of MCTP over SMBus.
Mgmt	**Title:** "DSP0238: Management Component Transport Protocol (MCTP) PCIe VDM Transport Binding Specification", Version 1.0.2, December 7, 2014 **Document:** http://dmtf.org/sites/default/files/standards/documents/DSP0238_1.0.1.pdf **Description:** Describes the binding of MCTP over PCIe.
Mgmt	**Title:** "DSP0239: Management Component Transport Protocol (MCTP) IDs and Codes", Version 1.2.0, August 28, 2012 **Document:** http://dmtf.org/sites/default/files/standards/documents/DSP0239_1.2.0.pdf **Description:** Describes constants used by MCTP specifications.
Mgmt	**Title:** "DSP0240: Platform Level Data Model (PLDM) Base Specification", Version 1.0.0, April 23, 2009 **Document:** http://www.dmtf.org/sites/default/files/standards/documents/DSP0240_1.0.0.pdf **Description:** PLDM base specification including the PLDM command format and PLDM control commands.
Mgmt	**Title:** "DSP2041: Platform Level Data Model (PLDM) over MCTP Binding Specification", Version 1.0.0, April 23, 2009 **Document:** http://www.dmtf.org/sites/default/files/standards/documents/DSP0241_1.0.0.pdf **Description:** Describes the binding of PLDM over MCTP.
Mgmt	**Title:** "DSP0245 Platform Level Data Model (PLDM) IDs and Codes Specification", Version 1.2.0, November 24, 2016 **Document:** http://www.dmtf.org/sites/default/files/standards/documents/DSP0245_1.2.0.pdf **Description:** Constants used for PLDM commands.
Mgmt	**Title:** "DSP0248: Platform Level Data Model (PLDM) for Platform Monitoring and Control Specification", Version 1.1.0, November 8, 2011 **Document:** http://www.dmtf.org/sites/default/files/standards/documents/DSP0248_1.1.0.pdf **Description:** Contains the description of PDT, sensors and affecters used in PLDM.
Mgmt	**Title:** "DSP0249: Platform Level Data Model (PLDM) State Set Specification", Version 1.0.0, March 16, 2009 **Document:** http://www.dmtf.org/sites/default/files/standards/documents/DSP0249_1.0.0.pdf **Description:** Defines the enums and states used by PLDM.
Mgmt	**Title:** "DSP0261: NC-SI Over MCTP Specification", Version 1.1.0, March 21, 2015 **Document:** http://www.dmtf.org/sites/default/files/standards/documents/DSP0261_1.1.0.pdf **Description:** Describes the encapsulation of NC-SI packets in MCTP.
Mgmt	**Title:** "DSP0267: Platform Level Data Model (PLDM) For Firmware Update Specification", Version 1.0.0, November 24, 2016 **Document:** http://www.dmtf.org/sites/default/files/standards/documents/DSP0267_1.0.0.pdf **Description:** Defines messages and data structures for updating firmware.
MPLS	**Title:** "RFC 3031: Multiprotocol Label Switching Architecture", January 2001 **Document:** https://datatracker.ietf.org/doc/rfc3031/ **Description:** MPLS architecture.
MPLS	**Title:** "RFC 3032: MPLS Label Stack Encoding", January 2001 **Document:** https://datatracker.ietf.org/doc/rfc3032/ **Description:** MPLS header.
PCI	**Title:** "PCI Local Bus Specification", Revision 3.0, February 3, 2004 **Document:** Available from www.pcisig.com **Description:** Compliant with select sections, such as Appendix I Vital Product Data.

Table 1-2. Standards Supported by the E810 [continued]

Category	Description
PCI	**Title:** "PCI Hot-Plug Specification", Revision 1.1, June 20, 2001 **Document:** Available from www.pcisig.com **Description:** Defines how PCI add-in cards are installed and removed while the system is running.
PCI	**Title:** "PCI Firmware Specification", Revision 3.0, June 20, 2005 **Document:** Available from www.pcisig.com **Description:** Defines the firmware interface for managing PCIe systems in a host computer. Describes the format, contents, and code entry points for Expansion ROMs.
PCI	**Title:** "PCI Express Base Specification", Revision 3.1a, December 7, 2015 **Document:** Available from www.pcisig.com **Description:** Contains the technical details of the PCIe architecture, protocol, link layer, physical layer, and software interface.
PCI	**Title:** "PCI Express Card Electromechanical Specification", Revision 3.0, July 21, 2013 **Document:** Available from www.pcisig.com **Description:** Mechanical and electrical specifications for an Evo card form factor.
Power Mgmt	**Title:** "Magic Packet Technology", November 1995 **Document:** http://support.amd.com/TechDocs/20213.pdf **Description:** Defines a method for waking up a sleeping networked PC using a specific Ethernet frame (a Magic packet).
Power Mgmt	**Title:** "PCI Bus Power Management Interface Specification", Revision 1.2, March 3, 2004 **Document:** Available from www.pcisig.com **Description:** Defines a standard set of PCI peripheral power management hardware interfaces and behavioral policies.
Power Mgmt	**Title:** "Advanced Configuration and Power Interface Specification", Revision 4.0a, April 5, 2010 **Document:** http://www.acpi.info/DOWNLOADS/ACPIspec40a.pdf **Description:** Defines standard interfaces to enable robust OS-directed motherboard device configuration and power management.
Power Mgmt	**Title:** "Standard ECMA-393: proxZZZy for sleeping hosts" 2nd Edition, June 2012 **Document:** www.ecma-international.org/publications/files/ECMA-ST/ECMA-393.pdf **Description:** Defines a low-power proxy that handles key network tasks for a high-power device, thus allowing the high-power device to sleep when not in active use.
RDMA	**Title:** "RFC 5040: A Remote Direct Memory Access Protocol Specification", October 2007 **Document:** http://www.ietf.org/rfc/rfc5040.txt **Description:** LLP is DDP, connects directly to the Host CPU/OS via the E810 System Interface.
RDMA	**Title:** "RFC 5041: Direct Data Placement over Reliable Transports", October 2007 **Document:** http://www.ietf.org/rfc/rfc5041.txt **Description:** LLP is MPA, ULP is RDMAP. DDP protocol provides information to place incoming data directly into a ULP receive buffer without intermediate buffers.
RDMA	**Title:** "RFC 5044: Marker PDU Aligned Framing for TCP Specification", October 2007 **Document:** http://www.ietf.org/rfc/rfc5044.txt **Description:** LLP is TCP, ULP is DDP. MPA is a framing protocol that enables the preservation of ULP record boundaries.
RDMA	**Title:** "RFC 7306: Remote Direct Memory Access (RDMA) Protocol Extensions", June 2014 **Document:** https://tools.ietf.org/html/rfc7306 **Description:** Specifies extensions to the IETF Remote Direct Memory Access Protocol (RDMAP) as specified in RFC 5040.
RDMA	**Title:** "RDMA Protocol Verbs Specification (Version 1.0)", April 2003 **Document:** rdmaconsortium.org/home/draft-hilland-iwarp-verbs-v1.0-RDMAC.pdf **Description:** Verbs describe the Host application/OS interface to an RNIC. This interface is implemented as a combination of the RNIC system interface, its associated firmware, and host software. It provides access to the RNIC queuing and memory management resources, as well as the underlying networking layers.

Table 1-2. Standards Supported by the E810 [continued]

Category	Description
RDMA	**Title:** "InfiniBand Architecture Specification Annex A17: RoCEv2" September 2, 2014 **Document:** Available from https://www.infinibandta.org/ibta-specifications-download/ **Description:** This document is an annex to Volume 1 release 1.2.1 of the InfiniBand Architecture.
RDMA	**Title:** "InfiniBand Architecture Specification Volume 1", Release 1.3, March 3, 2015 (subset of this spec that applies to RoCEv2) **Document:** Available from https://www.infinibandta.org/ibta-specifications-download/ **Description:** The InfiniBand Architecture Specification describes a first order interconnect technology for interconnecting processor nodes and I/O nodes to form a system area network.
Security	**Title:** "NIST SP 800-198: Platform Firmware Resiliency Guidelines", May 2018 **Abstract:** https://csrc.nist.gov/publications/detail/sp/800-193/final **Document:** https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-193.pdf **Description:** Provides technical guidelines and recommendations supporting resiliency of platform firmware and data against potentially destructive attacks.
Software	**Title:** Microsoft specification for "Receive Side Scaling" (RSS). **Document:** https://docs.microsoft.com/en-us/windows-hardware/drivers/network/ndis-receive-side-scaling2 **Description:** RSS is a network driver technology that enables the efficient distribution of network receive processing across multiple CPUs in multiprocessor systems.
SR-IOV	**Title:** "Single Root I/O Virtualization and Sharing Specification", Revision 1.1, January 20, 2010 **Document:** Available from www.pcisig.com **Description:** The SR-IOV specification defines extensions to the PCIe specification that enable the VMs in a virtualized server to efficiently share PCI adapter hardware resources.
Statistics	**Title:** "RFC 2819: Remote Network Monitoring Management Information Base", May 2000 **Document:** http://www.ietf.org/rfc/rfc2819.txt **Description:** Defines objects for managing remote network monitoring devices. Includes the popular packet size histogram counters. Obsoletes RFC 1757.
Statistics	**Title:** "RFC 2863: The Interfaces Group MIB", June 2000 **Document:** http://www.ietf.org/rfc/rfc2863.txt **Description:** Describes objects used for managing network interfaces. Obsoletes RFC 2233.
Statistics	**Title:** "RFC 4022: Management Information Base for the Transmission Control Protocol (TCP)", March 2005 **Document:** http://www.ietf.org/rfc/rfc4022.txt **Description:** IP version-independent TCP MIB. Obsoletes RFC 2012, 2452.
Statistics	**Title:** "RFC 4113: Management Information Base for the User Datagram Protocol (UDP)", June 2005 **Document:** http://www.ietf.org/rfc/rfc4113.txt **Description:** Objects for managing implementations of UDP. Obsoletes RFC 2013, 2454.
Statistics	**Title:** "RFC 4293: Management Information Base for the Internet Protocol (IP)", April 2006 **Document:** http://www.ietf.org/rfc/rfc4293.txt **Description:** IP version-independent IP MIB. Obsoletes RFC 2011, 2465, 2466.
Statistics	**Title:** "Microsoft NDIS 6.0 OID_GEN_STATISTICS" **Document:** https://docs.microsoft.com/en-us/windows-hardware/drivers/network/oid-gen-statistics **Description:** Adapter statistics counters defined by Microsoft for NDIS 6.0.
TCP	**Title:** "RFC 793: Transmission Control Protocol", September 1981 **Document:** http://www.ietf.org/rfc/rfc0793.txt **Description:** Base specification for TCP.
TCP	**Title:** "RFC 813: Window and Acknowledgment Strategy in TCP", July 1982 **Document:** http://www.ietf.org/rfc/rfc813.txt
TCP	**Title:** "RFC 896: Congestion Control in IP/TCP Internetworks", January 6, 1984 **Document:** http://www.ietf.org/rfc/rfc896.txt **Description:** Defines the Nagle algorithm, which specifies delaying transmission of small amounts of data when there are Acknowledgments (ACKs) outstanding.

Table 1-2. Standards Supported by the E810 [continued]

Category	Description
TCP	**Title:** "RFC 1323: TCP Extensions for High Performance", May 1992 **Document:** http://www.ietf.org/rfc/rfc1323.txt **Description:** Defines the TCP window scale and timestamp options, Round Trip Time Measurement (RTTM) and Protect Against Wrapped Sequences (PAWS). Obsoletes RFC 1072, 1185.
TCP	**Title:** "RFC 2525: Known TCP Implementation Problems", March 1999 **Document:** http://www.ietf.org/rfc/rfc2525.txt **Description:** This RFC describes implementation issues with various historical TCP/IP stacks. While it is included here for informative purposes only, it does serve as a good check list to avoid known problems. The E810 Protocol Engine is designed to support its recommendations.
TCP	**Title:** "RFC 2923: TCP Problems with Path MTU Discovery", September 2000 **Document:** http://www.ietf.org/rfc/rfc2923.txt **Description:** Discusses problems with existing RFC 1191 implementations that should be avoided. Serves as an implementation checklist.
TCP	**Title:** "RFC 2988: Computing TCP's Retransmission Timer", November 2000 **Document:** http://www.ietf.org/rfc/rfc2988.txt **Description:** Defines the standard algorithm that TCP senders are required to use to compute and manage their retransmission timer.
TCP	**Title:** "RFC 3390: Increasing TCP's Initial Window", October 2002 **Document:** http://www.ietf.org/rfc/rfc3390.txt **Description:** Specifies an optional standard for TCP to increase the permitted initial window from one or two segments to roughly 4 KB. Obsoletes RFC 2414. Updates RFC 2581.
TCP	**Title:** "RFC 3465: TCP Congestion Control with Appropriate Byte Counting (ABC)", February 2003 **Document:** http://www.ietf.org/rfc/rfc3465.txt **Description:** This experimental RFC defines a small modification to the way TCP increases its congestion window. Instead of increasing cwnd by a constant amount for each acknowledgment, cwnd is increased based on the number of previously unacknowledged bytes each ACK covers.
TCP	**Title:** "RFC 3782: The NewReno Modification to TCP's Fast Recovery Algorithm", April 2004 **Document:** http://www.ietf.org/rfc/rfc3782.txt **Description:** Update the fast recovery algorithm that is run after three duplicate ACKs have been received. Changes to the CWND during fast recovery, scheduling additional fast retransmitted packets if the received ACKs do not acknowledge all the data when fast recovery was entered. Obsoletes RFC 2582.
TCP	**Title:** "RFC 5681: TCP Congestion Control", September 2009 **Document:** http://www.ietf.org/rfc/rfc5681.txt **Description:** TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. Obsoletes RFC 2581.
Test	**Title:** "IEEE Standard Test Access Port and Boundary-Scan Architecture" (IEEE Std 1149.1-2001) **Document:** Available from http://standards.ieee.org/getieee802 **Description:** Defines a standard interface through which instructions and test data are communicated to an integrated circuit for component- and circuit board-level testing.
Tunneling	**Title:** "RFC 7637: NVGRE: Network Virtualization using Generic Routing Encapsulation", September 2015 **Document:** http://datatracker.ietf.org/doc/draft-sridharan-virtualization-nvgre/?include_text=1 **Description:** MAC in Generic Routing Encapsulation (GRE) over IP encapsulation.
Tunneling	**Title:** "RFC 7348: Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", August 2014 **Document:** http://datatracker.ietf.org/doc/draft-mahalingam-dutt-dcops-vxlan/?include_text=1 **Description:** MAC in UDP encapsulation.
UDP	**Title:** "RFC 768: User Datagram Protocol", August 28, 1980 **Document:** http://www.ietf.org/rfc/rfc0768.txt **Description:** Base specification for UDP.

Chapter 2 Pin Interface

2.1 Pin Descriptions

This section provides detailed descriptions of E810 signal pins, grouped by function.

- A "_N" following the signal name indicates that the signal is active-low.
- Signal names with a suffix of "_p" and "_n" refer to differential signals.

The buffer types are listed in Table 2-1.

Table 2-1. Buffer Types

Buffer	Description
In	Input is a standard input-only signal.
Out (O)	Totem Pole Output (TPO) is a standard active driver.
t/s	Tri-state is a bi-directional, tri-state input/output pin.
o/d	Open drain enables multiple devices to share as a wire-OR.
A-in	Analog input signals.
A-out	Analog output signals.
A-Inout	Bi-directional analog signals.
HCSL-in	High-Speed Current Steering Logic input signal.
NCSI-in	NC-SI input signal.
NCSI-out	NC-SI output signal.
Pu	Internal pull-up resistor.
Pd	Internal pull-down resistor.
clkobs	Clock observation output.
JTAG	1.8 V JTAG I/O.

2.2 Pin Assignments and Descriptions

The E810 is available in two package options:

- The 25x25 mm, 668-pin, Flip-Chip Ball Grid Array (FCBGA) package contains all the signals needed to implement the maximum product configuration. In particular, 16 lanes of PCIe and eight lanes of Ethernet PMD are exposed. Products based on the Intel® Ethernet Controller E810-CAM1 and the Intel® Ethernet Controller E810-CAM2 use this package.
- The 21x21 mm, 456-pin, Flip-Chip Ball Grid Array (FCBGA) package contains the signals needed to implement eight lanes of PCIe and two lanes of Ethernet PMD for the dual-port 25 GbE or single-port 50 GbE implementation. Products based on the Intel® Ethernet Controller E810-XXVAM2 use this package.

The following sections provide the signal names, pin/ball assignments and signal descriptions. The AC/DC electrical specifications for the signals are described in Chapter 16.

2.2.1 PCIe Interface Pins

This section provides the pin assignment for PCIe interface signals.

Table 2-2. PCIe Interface Pins

Signal	Ball # CAM2/CAM1	Ball # XXVAM2	Type	Description
PE_CLK_p PE_CLK_n	N47 N45	H40 G39	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.
PET_0_p PET_0_n	L39 L41	E39 E37	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	J39 J41	C35 C33	A-out	Same as previous.
PET_2_p PET_2_n	G39 G41	C29 C27	A-out	Same as previous.
PET_3_p PET_3_n	E45 D44	C23 C21	A-out	Same as previous.
PET_4_p PET_4_n	E41 D40	C17 C15	A-out	Same as previous.
PET_5_p PET_5_n	E37 D36	E11 D10	A-out	Same as previous.
PET_6_p PET_6_n	E33 D32	E7 D6	A-out	Same as previous.
PET_7_p PET_7_n	E29 D28	E3 D2	A-out	Same as previous.
PET_8_p PET_8_n	E25 D24	N/A	A-out	Same as previous.
PET_9_p PET_9_n	E21 D20	N/A	A-out	Same as previous.
PET_10_p PET_10_n	E17 D16	N/A	A-out	Same as previous.
PET_11_p PET_11_n	E13 D12	N/A	A-out	Same as previous.
PET_12_p PET_12_n	E9 D8	N/A	A-out	Same as previous.
PET_13_p PET_13_n	E5 D4	N/A	A-out	Same as previous.
PET_14_p PET_14_n	G9 G7	N/A	A-out	Same as previous.
PET_15_p PET_15_n	J9 J7	N/A	A-out	Same as previous.
PER_0_p PER_0_n	L45 L47	C39 B38	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 output carries both data and an embedded clock that is recovered along with data at the receiving end.
PER_1_p PER_1_n	J45 J47	A35 A33	A-in	Same as previous.
PER_2_p PER_2_n	G45 G47	A29 A27	A-in	Same as previous.

Table 2-2. PCIe Interface Pins [continued]

Signal	Ball # CAM2/CAM1	Ball # XXVAM2	Type	Description
PER_3_p PER_3_n	B44 A43	A23 A21	A-in	Same as previous.
PER_4_p PER_4_n	B40 A39	A17 A15	A-in	Same as previous.
PER_5_p PER_5_n	B36 A35	B12 A11	A-in	Same as previous.
PER_6_p PER_6_n	B32 A31	B8 A7	A-in	Same as previous.
PER_7_p PER_7_n	B28 A27	B4 A3	A-in	Same as previous.
PER_8_p PER_8_n	B24 A23	N/A	A-in	Same as previous.
PER_9_p PER_9_n	B20 A19	N/A	A-in	Same as previous.
PER_10_p PER_10_n	B16 A15	N/A	A-in	Same as previous.
PER_11_p PER_11_n	B12 A11	N/A	A-in	Same as previous.
PER_12_p PER_12_n	B8 A7	N/A	A-in	Same as previous.
PER_13_p PER_13_n	B4 A3	N/A	A-in	Same as previous.
PER_14_p PER_14_n	G3 G1	N/A	A-in	Same as previous.
PER_15_p PER_15_n	J3 J1	N/A	A-in	Same as previous.
PE_WAKE_N	R47	Y40	o/d	Wake. Pulled to 0b to indicate that a Power Management Event (PME) is pending and the PCIe link should be restored. Defined in the PCIe specifications. 3.3 V tolerant even when device is not powered.
PE_RST_N	R45	W39	In	Power and Clock Good Indication. Indicates that power and PCIe reference clock are within specified values. Defined in the PCIe specifications. Also called PCIe Reset and PERST#.

2.2.2 Ethernet Interface Pins

This section provides the pin assignments for Ethernet interface signals.

Table 2-3. Ethernet Interface Pins

Signal	Ball # CAM2/CAM1	Ball # XXVAM2	Type	Description
REFCLKIN_p REFCLKIN_n	M2 M4	G1 H2	HCSL-in	156.25MHz HCSL differential clock input
RX_L0_p RX_L0_n	AF44 AE45	AA17 AA19	A-in	Serial Data Input for Ethernet PMD lane 0. A serial differential input pair running at up to 26.5625 GBaud. An embedded clock present in this input is recovered along with the data. This lane is used as a receive pair for one of the Ethernet ports. See Section 3.2.2 for lane to port mapping
RX_L1_p RX_L1_n	AF38 AF40	AA23 AA25	A-in	Same as previous, Lane 1.
RX_L2_p RX_L2_n	AF32 AF34	N/A	A-in	Same as previous, Lane 2.
RX_L3_p RX_L3_n	AF26 AF28	N/A	A-in	Same as previous, Lane 3.
RX_L4_p RX_L4_n	AF20 AF22	N/A	A-in	Same as previous, Lane 4.
RX_L5_p RX_L5_n	AF14 AF16	N/A	A-in	Same as previous, Lane 5.
RX_L6_p RX_L6_n	AF8 AF10	N/A	A-in	Same as previous, Lane 6.
RX_L7_p RX_L7_n	AF2 AF4	N/A	A-in	Same as previous, Lane 7.
TX_L0_p TX_L0_n	AG47 AF48	AC17 AC19	A-out	Serial Data Output for Ethernet PMD Lane 0. A serial differential output pair running at up to 26.5625 GBaud. This output carries both data and an embedded clock that is recovered along with data at the receiving end. This lane is used as a transmit pair for one of the Ethernet ports for one of the Ethernet ports. See Section 3.2.2 for lane to port mapping
TX_L1_p TX_L1_n	AH42 AH44	AC23 AC25	A-out	Same as previous, Lane 1.
TX_L2_p TX_L2_n	AH36 AH38	N/A	A-out	Same as previous, Lane 2.
TX_L3_p TX_L3_n	AH30 AH32	N/A	A-out	Same as previous, Lane 3.
TX_L4_p TX_L4_n	AH24 AH26	N/A	A-out	Same as previous, Lane 4.
TX_L5_p TX_L5_n	AH18 AH20	N/A	A-out	Same as previous, Lane 5.
TX_L6_p TX_L6_n	AH12 AH14	N/A	A-out	Same as previous, Lane 6.
TX_L7_p TX_L7_n	AH6 AH8	N/A	A-out	Same as previous, Lane 7.

Table 2-4. External Ethernet PHY Control - MDIO/I²C Interface Pins

Signal	Ball # CAM2/CAM1	Ball # XXVAM2	Type	Description
MDIO0_SDA0	AC45	AC33	T/s, o/d, Pu	Management Data, when configured as an MDIO interface. Bi-directional signal for serial data transfers between the E810 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 netlist stored in NVM.
MDC0_SCL0	AB38	W33	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 netlist stored in NVM. ***Note:*** This I/O operates as an open drain buffer, and therefore requires an external pull-up device.
MDIO1_SDA1	AB42	AA33	T/s, o/d, Pu	Same as above.
MDC1_SCL1	Y38	R31	O, o/d, Pu	Same as above
MDIO2_SDA2	AB48	AC37	T/s, o/d, Pu	Same as above.
MDC2_SCL2	AA39	U33	O, o/d, Pu	Same as above
MDIO3_SDA3	AA45	AA37	T/s, o/d, Pu	Same as above.
MDC3_SCL3	AB40	Y34	O, o/d, Pu	Same as above
MDIO4_SDA4	Y48	Y38	T/s, o/d, Pu	Same as above.
MDC4_SCL4	Y40	V34	O, o/d, Pu	Same as above

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.

2.2.3 NC-SI Interface Pins

This section provides the pin assignment for NC-SI signals.

Table 2-5. NC-SI Interface Pins

Signal	Ball # CAM2/CAM1	Ball # XXVAM2	Type	Description
NCSI_CLK_IN	U39	U37	NCSI-In, Pu	NC-SI Reference Clock Input. Synchronous clock reference for receive, transmit, and control interface. It is a 50 MHz clock ±100 ppm.
NCSI_CRS_DV	U45	T40	NCSI-Out, Pu	Carrier Sense/Receive Data Valid (CRS/DV).
NCSI_RXD_0 NCSI_RXD_1	V46 Y42	U39 N33	NCSI-Out, Pu	Receive Data. Data signals to the BMC.
NCSI_TX_EN	V42	V40	NCSI-In, Pu	Transmit Enable.
NCSI_TXD_0 NCSI_TXD_1	V40 W41	V38 T34	NCSI-In, Pu	Transmit Data. Data signals from the BMC.
NCSI_ARB_IN	U41	T38	NCSI-In, Pu	NC-SI Hardware Arbitration Input. If GL_MNG_HWARB_CTRL.*NCSI_ARB_EN* is cleared, this pin is internally pulled up.
NCSI_ARB_OUT	Y46	P34	NCSI-Out, Pu	NC-SI Hardware Arbitration Output.

Note: If NC-SI is disconnected, external pull-downs should be used for the NCSI_CLK_IN, NCSI_TXD[1:0], and NCSI_TX_EN signals.

2.2.4 SMBus Interface Pins

This section provides the pin assignment for the SMBus interface signals.

Table 2-6. SMBus Interface Pins

Signal	Ball # CAM2/CAM1	Ball # XXVAM2	Type	Description
SMBCLK	AC41	AA31	o/d	SMBus Clock. One clock pulse is generated for each data bit transferred. 3.3 V tolerant when device is not powered.
SMBD	AA37	W31	o/d	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	AC39	AC31	o/d	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, an external pull-up should be used for the SMBCLK and SMBD pins.

2.2.5 Serial Flash Memory Interface Pins

This section provides the pin assignment for SPI signals for connectivity to Flash memory devices.

Table 2-7. Serial Flash Memory Interface Pins

Signal	Ball # CAM2/CAM1	Ball # XXVAM2	Type	Description
FLSH_SI	W1	U1	t/s, Pu	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 4-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	AA1	W1	In, Pu	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 4-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	Y2 U1	V2 R1	t/s, Pu	FLSH_IO2 and FLSH_IO3 are only used for quad SPI instructions, providing a 4-bit wide data path to the Flash device.
FLSH_SCK	V2	T2	t/s, Pu	Flash serial clock operates at 50 MHz.
FLSH_CE_N	AB2	Y2	t/s, Pu	Flash chip select output.

2.2.6 General Purpose I/O (GPIO) Pins

This section provides the pin assignment for GPIO signals. The E810 has a total of 24 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 E810 offers the flexibility to configure any of the GPIO pins to different modes and associated with different ports as described in Section 3.5.

The GPIO pins fall into two categories:

- **GPIO/SDP/LED:** These general purpose software definable pins are driven via the internal I/O expander (the 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 netlist. Refer to the reference design section for supported mapping.
- **Direct Timing GPIO:** The E810 has a total of five direct, low latency, GPIOs that can be used for accurate timing applications such as 1588. Four of these GPIOs are single ended, one a differential output. These pins are controlled directly by the controller core to minimize delay and jitter for timing applications.

Table 2-8. GPIO/SDP/LED Interface Pins

Signal	Ball # CAM2/CAM1	Ball # XXVAM2	Type	Description
SDP0 SDP1 SDP2 SDP3 SDP4 SDP5 SDP6 SDP7 SDP8 SDP9 SDP10 SDP11 SDP12 SDP13 SDP14 SDP15 SDP16 SDP17 SDP18 SDP19	V4 T10 U3 U7 U11 W3 Y4 V10 U9 W7 W9 V8 Y8 W11 AA3 Y10 AA7 AB4 V38 W39	T4 P6 R3 V4 R7 U3 AA1 U7 W3 AA3 V8 Y4 AB4 R9 AC3 U9 AC5 AB6 W37 R33	t/s, Pu	General purpose 3.3 V I/Os. Can be used to connect LEDs, low speed optical module interfaces, external PHY control, or other similar usages. The SDP pins can also be configured for use as external interrupt sources. Refer to the reference design section for supported uses.

Table 2-9. Direct Timing GPIO Interface Pins

Signal	Ball # CAM2/CAM1	Ball # XXVAM2	Type	Description
SDP20 SDP21 SDP22 SDP23	AC1 AA9 AB8 AC3	AA9 W7 AA7 AC9	t/s, Pu	General purpose 3.3 V I/Os for 1588 synchronized timing applications. The pins can also be configured for use as external interrupt sources.
CLK_OUT_P CLK_OUT_N	P4 P2	M2 L1	clkobs	0.8 V differential output for 1588 synchronized timing applications.

2.2.7 Miscellaneous Pins

This section provides the pin assignment for other miscellaneous signals.

Table 2-10. Miscellaneous Pins

Signal	Ball # CAM2/CAM1	Ball # XXVAM2	Type	Description
LAN_PWR_GOOD	AA11	Y10	In, Pu	LAN Power Good. A 3.3 V input signal. A transition from low-to-high initializes the E810 into operation. If not used (POR_BYPASS = 0b), an internal Power-on-Reset (POR) circuit triggers the E810 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	AD2	AB10	In, Pu	Bypass indication as to whether or not to use the internal POR or the LAN_PWR_GOOD pin. 0b = The E810 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 Chapter 16 could not be met. 1b = The E810 disables the internal POR circuit and uses the LAN_PWR_GOOD pin as a POR indication.
AUX_PWR	AD4	AB12	In, Pu	Auxiliary Power Available. When set, indicates that auxiliary power is available and the E810 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	AC7	AC11	In, Pu	Main Power OK. Indicates that platform main power is up. Must be connected externally.
PCI_DIS_N	AB10	Y8	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, all PCI functions as configured from NVM are enabled. If this pin is asserted/driven low during initialization, all PCI functions that are allowed to be disabled as configured in NVM are disabled (see Section 4.5 for details).
DEV_DIS_N	AC9	W9	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 (see Section 4.5 for details). If this pin is not connected or driven high during initialization, 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, 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).
SENSOR_THERM_IN SENSOR_THERM_OUT	M10 M8	J7 J5	Analog	Thermal Diode for use with the integrated thermal sensor. Use with an external thermal sensor is not validated.
OBS_CORE_VDD OBS_CORE_VSS	T24 R23	P22 N21		Power supply observation pins.

2.2.8 Testability and Debug Pins

This section provides the pin assignment for JTAG testability interface signals.Note that the JTAG 1.8 V I/Os internal pull-up/pull-down are not enabled until after core VDD is applied. Therefor, for stable power up, external pull-up/pull-down resistors are required as indicated. Access to the JTAG pins is need for advanced debug. Refer to Section 17, "Design Guidelines" for recommended debug connections.

Table 2-11. Testability and Debug Pins

Signal	Ball # CAM2/CAM1	Ball # XXVAM2	Type	Description
JTCK	R41	K38	JTAG, In, Pu	JTAG Clock Input. Connect pull up resistor to 1.8V for normal operation.
JTDI	P38	K36	JTAG, In, Pu	JTAG Data Input. Connect pull up resistor to 1.8V for normal operation.
JTDO	P42	J37	JTAG, Out	JTAG Data Output. 1.8 V
JTMS	N39	K34	JTAG, In, Pu	JTAG TMS Input. Connect pull up resistor to 1.8 V for normal operation.
JRST_N	R37	L37	JTAG, In, Pd	JTAG Reset Input. 1.8 V I/O. Active low reset for the JTAG port. Connect pull down resistor to VSS for normal operation.
TAP_SEL	P40	L39	JTAG, In, Pd	1.8V I/O. 0b = JTAG interface is connected to Intel TAP. 1b = JTAG interface is connected to manufacturing TAP.
UARTTXD	AA47	AA39	t/s, Pu	Multi-function pin. RSVD pins. UART interface for EMP firmware debug. Transmit data output. This pin can also be configured as an additional SDP from the I/O Widget (IOW).
UARTRXD	AA41	AB38	t/s, Pu	Multi-function pin. RSVD pins. UART interface for EMP firmware debug. Receive data input. This pin can also be configured as an additional SDP from the IOW.
UARTCTS	AB46	Y32	t/s, Pd	Multi-function pin. RSVD pins. UART interface for EMP firmware debug. Clear to send input. This pin can also be configured as an additional SDP from the IOW.
UARTRTS	AC47	AB34	t/s, Pu	Multi-function pin. RSVD pins. UART interface for EMP firmware debug. Request to send output. This pin can also be configured as an additional SDP from the IOW.

2.2.9 Reserved and No-Connect Pins

This section provides the pin assignment for reserved and no-connect pins. Table 2-12 and Table 2-13 contain the reserved and no-connect pins for the E810-CAM2/CAM1 and E810-XXVAM2, respectively.

Table 2-12. Reserved and No-Connect Pins (E810-CAM2/CAM1)

Signal	Ball # CAM2/CAM1	Description
RSVDN37_VDDH RSVDR39_VDDH	N37 R39	Reserved pins. Connect pull up resistor to 1.8 V for normal operation.
RSVDN41_VSS	N41	Reserved pin. Connect pull down resistor to VSS for normal operation.
RSVDN7_NC RSVDN11_NC RSVDP8_NC RSVDP10_NC RSVDR7_NC RSVDR9_NC RSVDR11_NC RSVDT2_NC RSVDT4_NC RSVDT8_NC RSVDT38_NC RSVDT40_NC RSVDT42_NC RSVDT46_NC RSVDT48_NC RSVDU47_NC RSVDV48_NC RSVDW33_NC RSVDW45_NC RSVDW47_NC RSVDAA35_NC	N7 N11 P8 P10 R7 R9 R11 T2 T4 T8 T38 T40 T42 T46 T48 U47 V48 W33 W45 W47 AA35	Reserved pins. Leave unconnected for normal operation.

Table 2-13. Reserved and No-Connect Pins (E810-XXVAM2)

Signal	Ball # XXVAM2	Description
RSVDJ33_VDDH RSVDM40_VDDH	J33 M40	Reserved pins. Connect pull up resistor to 1.8 V for normal operation.
RSVDK40_VSS	K40	Reserved pin. Connect pull down resistor to VSS for normal operation.
RSVDH6_NC RSVDK4_NC RSVDK6_NC RSVDL5_NC RSVDL7_NC RSVDL33_NC RSVDM6_NC RSVDM34_NC RSVDM38_NC RSVDN5_NC RSVDN7_NC RSVDN9_NC RSVDN37_NC RSVDN39_NC RSVDP4_NC RSVDP38_NC RSVDP40_NC RSVDR13_NC RSVDR37_NC RSVDR39_NC RSVDAB40_NC	H6 K4 K6 L5 L7 L33 M6 M34 M38 N5 N7 N9 N37 N39 P4 P38 P40 R13 R37 R39 AB40	Reserved pins. Leave unconnected for normal operation.

2.2.10 Power Supply Pins

This section provides the pin assignment for power supply pins. The electrical specifications for the power supply pins are defined in Section 16.3. Table 2-14 and Table 2-15 contain the power supply pins for the E810-CAM2/CAM1 and E810-XXVAM2, respectively.

Table 2-14. Power Supply Pins (E810-CAM2/CAM1)

Signal	Ball # CAM2/CAM1	Type	Description
VDDIO33	P12, P14, T12, T14, T34, T36, V12, V14, V34, V36, Y12, Y14, Y34, Y36	3.3 V	Digital power supply for 3.3 V I/O.
VDDH18	M12, M14, P34, P36	1.8 V	Digital 1.8 V power supply.
AVDDH	Y20, Y22, Y24	1.1 V	Analog power supply for Ethernet SerDes interfaces.
AVDD_ETH	AB14, AB16, AB18, AB20, AB22, AB24, AB26, AB28, AB30, AB32, AB34, AD12, AD14, AD16, AD18, AD20, AD22, AD24, AD26, AD28, AD30, AD32, AD34, AD36	0.9 V	Analog power supply for Ethernet SerDes interfaces.
AVDD_PCIE	F14, F16, F18, F20, F22, F24, F26, F28, F30, F32, F34, H16, H18, H20, H22, H24, H26, H28, H30, H32, K22, K24, K26, K28, K30	0.9 V	Analog power supply for PCIe SerDes interfaces.
AVDD_PLL	AA19, AA23	0.9 V	PLL power supply.
VDD	K20, K32, M16, M18, M20, M22, M24, M26, M28, M30, M32, P16, P18, P20, P22, P24, P26, P28, P30, P32, T16, T18, T20, T22, T26, T28, T30, T32, V16, V18, V20, V22, V24, V26, V28, V30, V32, Y16, Y18, Y26, Y28, Y30, Y32	0.8 V	Digital core logic power supply.
PLLVDD	J17, K18	0.8 V	
VSSA	A5, A9, A13, A17, A21, A25, A29, A33, A37, A41, A45, B2, B6, B10, B14, B18, B22, B26, B30, B34, B38, B42, B46, B48, C1, C3, C5, C7, C9, C11, C13, C15, C17, C19, C21, C23, C25, C27, C29, C31, C33, C35, C37, C39, C41, C43, C45, C47, D2, D6, D10, D14, D18, D22, D26, D30, D34, D38, D42, D46, D48, E1, E3, E7, E11, E15, E19, E23, E27, E31, E35, E39, E43, E47, F2, F4, F6, F8, F10, F12, F36, F38, F40, F42, F44, F46, F48, G5, G11, G13, G15, G17, G19, G21, G23, G25, G27, G29, G31, G33, G35, G37, G43, H2, H4, H6, H8, H10, H12, H14, H34, H36, H38, H40, H42, H44, H46, H48, J5, J11, J13, J15, J21, J23, J25, J27, J29, J31, J33, J35, J37, J43, K2, K4, K6, K8, K10, K12, K34, K36, K38, K40, K42, K44, K46, K48, L35, L37, L43, M34, M36, M38, M40, M42, M44, M46, M48, W21, W23, AA15, AA17, AA21, AA25, AA27, AA29, AA31, AA33, AB12, AB36, AC11, AC13 AC15, AC17, AC19, AC21, AC23, AC25, AC27, AC29, AC31, AC33, AC35, AC37, AD8, AD10, AD38, AD40, AD42, AD44, AD46, AD48, AE1, AE3, AE5, AE7, AE9, AE11, AE13, AE15, AE17, AE19, AE21, AE23, AE25, AE27, AE29, AE31, AE33, AE35, AE37, AE39, AE41, AE43, AE47, AF6, AF12, AF18, AF24, AF30, AF36, AF42, AF46, AG1, AG3, AG5, AG7, AG9, AG11, AG13, AG15, AG17, AG19, AG21, AG23, AG25, AG27, AG29, AG31, AG33, AG35, AG37, AG39, AG41, AG43, AG45, AH4, AH10, AH16, AH22, AH28, AH34, AH40, AH46	0 V	Analog power supply ground.
PLLGND	J19, K16	0 V	
VSS	K14, L1, L3, L5, L7, L9, L11, L13, L15, L17, L19, L21, L23, L25, L27, L29, L31, L33, M6, N1, N3, N5, N9, N13, N15, N17, N19, N21, N23, N25, N27, N29, N31, N33, N35, N43, P6, P44, P46, P48, R1, R3, R5, R13, R15, R17, R19, R21, R25, R27, R29, R31, R33, R35, R43, T6, T44, U5, U13, U15, U17, U19, U21, U23, U25, U27, U29, U31, U33, U35, U37, U43, V6, V44, W5, W13, W15, W17, W19, W25, W27, W29, W31, W35, W37, W43, Y6, Y44, AA5, AA13, AA43, AB6, AB44, AC5, AC43, AD6	0 V	Digital power supply ground.

Table 2-15. Power Supply Pins (E810-XXVAM2)

Signal	Ball # XXVAM2	Type	Description
VDDIO33	M8, M10, M12, M30, M32, P8, P10, P12, P30, P32, T8, T10, T30, T32, V30, V32	3.3 V	Digital power supply for 3.3 V I/O.
VDDH18	H8, H10, H30, H32, K8, K10, K30, K32	1.8 V	Digital 1.8 V power supply.
AVDD	D14, D16, D18, D20, D22, D24, D26, D28, F18, F20, F22, F24, F26, F28, F30, U19, U21, W13, W15, W17, W19, W21, W23, W25, W27	0.9 V	Analog power supply for PCIe and Ethernet SerDes interfaces.
VDD	H12, H18, H20, H22, H24, H26, H28, K12, K14, K16, K18, K20, K22, K24, K26, K28, M14, M16, M18, M20, M22, M24, M26, M28, P14, P16, P18, P20, P24, P26, P28, T12, T14, T16, T18, T22, T24, T26, T28	0.8 V	Digital core logic power supply.
PLLVDD	G15, H16	0.8 V	
VSSA	A5, A9, A13, A19, A25, A31, A37, B2, B6, B10, B14, B16, B18, B20, B22, B24, B26, B28, B30, B32, B34, B36, B40, C1, C3, C5, C7, C9, C11, C13, C19, C25, C31, C37, D4, D8, D12, D30, D32, D34, D36, D38, D40, E1, E5, E9, E13, E15, E17, E19, E21, E23, E25, E27, E29, E31, E33, E35, F2, F4, F6, F8, F10, F32, F34, F36, F38, F40, G19, G21, G23, T20, V12, V14, V16, V18, V20, V22, V24, V26, V28, Y14, Y16, Y18, Y20, Y22, Y24, Y26, Y28, AA13, AA15, AA21, AA27, AB14, AB16, AB18, AB20, AB22, AB24, AB26, AB28, AC15, AC21, AC27	0 V	Analog power supply ground.
PLLGND	G17, H14	0 V	
VSS	F12, F14, F16, G3, G5, G7, G9, G11, G13, G25, G27, G29, G31, G33, G35, G37, H4, H34, H36, H38, J1, J3, J9, J11, J13, J15, J17, J19, J21, J23, J25, J27, J29, J31, J35, J39, K2, L3, L9, L11, L13, L15, L17, L19, L21, L23, L25, L27, L29, L31, L35, M4, M36, N1, N3, N11, N13, N15, N17, N19, N23, N25, N27, N29, N31, N35, P2, P36, R5, R11, R15, R17, R19, R21, R23, R25, R27, R29, R35, T6, T36, U5, U11, U13, U15, U17, U23, U25, U27, U29, U31, U35, V6, V10, V36, W5, W11, W29, W35, Y6, Y12, Y30, Y36, AA5, AA11, AA29, AA35, AB2, AB8, AB30, AB32, AB36, AC7, AC13, AC29, AC35	0 V	Digital power supply ground.

2.2.11 Pull-Up and Pull-Down Resistors

Internal pull-up and pull-down values on 3.3 V I/O pins, where indicated:

- Min: 10 KΩ
- Max: 20 KΩ

Internal pull-up and pull-down values on 1.8 V JTAG I/O pins, where indicated:

- Min: 25 KΩ
- Max: 50 KΩ

2.3 Package Layout

Figure 2-1 and Figure 2-2 show a top view ball map of Intel® Ethernet Controller E810-CAM2/CAM1. Figure 2-3 and Figure 2-4 show a top view ball map of Intel® Ethernet Controller E810-XXVAM2. See Section 16.7 for package mechanical specifications.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
A			PER_13_n		VSSA		PER_12_n		VSSA		PER_11_n		VSSA		PER_10_n		VSSA		PER_9_n		VSSA		PER_8_n	
B		VSSA		PER_13_p		VSSA		PER_12_p		VSSA		PER_11_p		VSSA		PER_10_p		VSSA		PER_9_p		VSSA		PER_8_p
C	VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA	
D		VSSA		PET_13_n		VSSA		PET_12_n		VSSA		PET_11_n		VSSA		PET_10_n		VSSA		PET_9_n		VSSA		PET_8_n
E	VSSA		VSSA		PET_13_p		VSSA		PET_12_p		VSSA		PET_11_p		VSSA		PET_10_p		VSSA		PET_9_p		VSSA	
F		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		AVDD_PCIE		AVDD_PCIE		AVDD_PCIE		AVDD_PCIE		AVDD_PCIE		AVDD_PCIE
G	PER_14_n		PER_14_p		VSSA		PET_14_n		PET_14_p		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA	
H		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		AVDD_PCIE		AVDD_PCIE		AVDD_PCIE		AVDD_PCIE		AVDD_PCIE
J	PER_15_n		PER_15_p		VSSA		PET_15_n		PET_15_p		VSSA		VSSA		VSSA		PLLVDD		PLLGND		VSSA		VSSA	
K		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSS		PLLGND		PLLVDD		VDD		AVDD_PCIE		AVDD_PCIE
L	VSS		VSS		VSS		VSS		VSS		VSS		VSS		VSS		VSS		VSS		VSS		VSS	
M		REFCLKIN_p		REFCLKIN_n		VSS		SENSOR_THERM_OUT		SENSOR_THERM_IN		VDDH18		VDDH18		VDD		VDD		VDD		VDD		VDD
N	VSS		VSS		VSS		RSVDN7_NC		VSS		RSVDN11_NC		VSS		VSS		VSS		VSS		VSS		VSS	
P		CLK_OUT_N		CLK_OUT_P		VSS		RSVDP8_NC		RSVDP10_NC		VDDIO33		VDDIO33		VDD		VDD		VDD		VDD		VDD
R	VSS		VSS		VSS		RSVDR7_NC		RSVDR9_NC		RSVDR11_NC		VSS		VSS		VSS		VSS		VSS		OBS_CORE_VSS	
T		RSVDT2_NC		RSVDT4_NC		VSS		RSVDT8_NC		SDP1		VDDIO33		VDDIO33		VDD		VDD		VDD		VDD		OBS_CORE_VDD
U	FLSH_IO3		SDP2		VSS		SDP3		SDP8		SDP4		VSS		VSS		VSS		VSS		VSS		VSS	
V		FLSH_SCK		SDP0		VSS		SDP11		SDP7		VDDIO33		VDDIO33		VDD		VDD		VDD		VDD		VDD
W	FLSH_SI		SDP5		VSS		SDP9		SDP10		SDP13		VSS		VSS		VSS		VSS		VSSA		VSSA	
Y		FLSH_IO2		SDP6		VSS		SDP12		SDP15		VDDIO33		VDDIO33		VDD		VDD		AVDDH		AVDDH		AVDDH
AA	FLSH_SO		SDP14		VSS		SDP16		SDP21		LAN_PWR_GOOD		VSS		VSSA		VSSA		AVDD_PLL		VSSA		AVDD_PLL	
AB		FLSH_CE_N		SDP17		VSS		SDP22		PCI_DIS_N		VSSA		AVDD_ETH		AVDD_ETH		AVDD_ETH		AVDD_ETH		AVDD_ETH		AVDD_ETH
AC	SDP20		SDP23		VSS		MAIN_PWR_OK		DEV_DIS_N		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA	
AD		POR_BYPASS		AUX_PWR		VSS		VSSA		VSSA		AVDD_ETH		AVDD_ETH		AVDD_ETH		AVDD_ETH		AVDD_ETH		AVDD_ETH		AVDD_ETH
AE	VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA	
AF		RX_L7_p		RX_L7_n		VSSA		RX_L6_p		RX_L6_n		VSSA		RX_L5_p		RX_L5_n		VSSA		RX_L4_p		RX_L4_n		VSSA
AG	VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA	
AH				VSSA		TX_L7_p		TX_L7_n		VSSA		TX_L6_p		TX_L6_n		VSSA		TX_L5_p		TX_L5_n		VSSA		TX_L4_p

Figure 2-1. E810-CAM2/CAM1 Package Layout (Part 1)

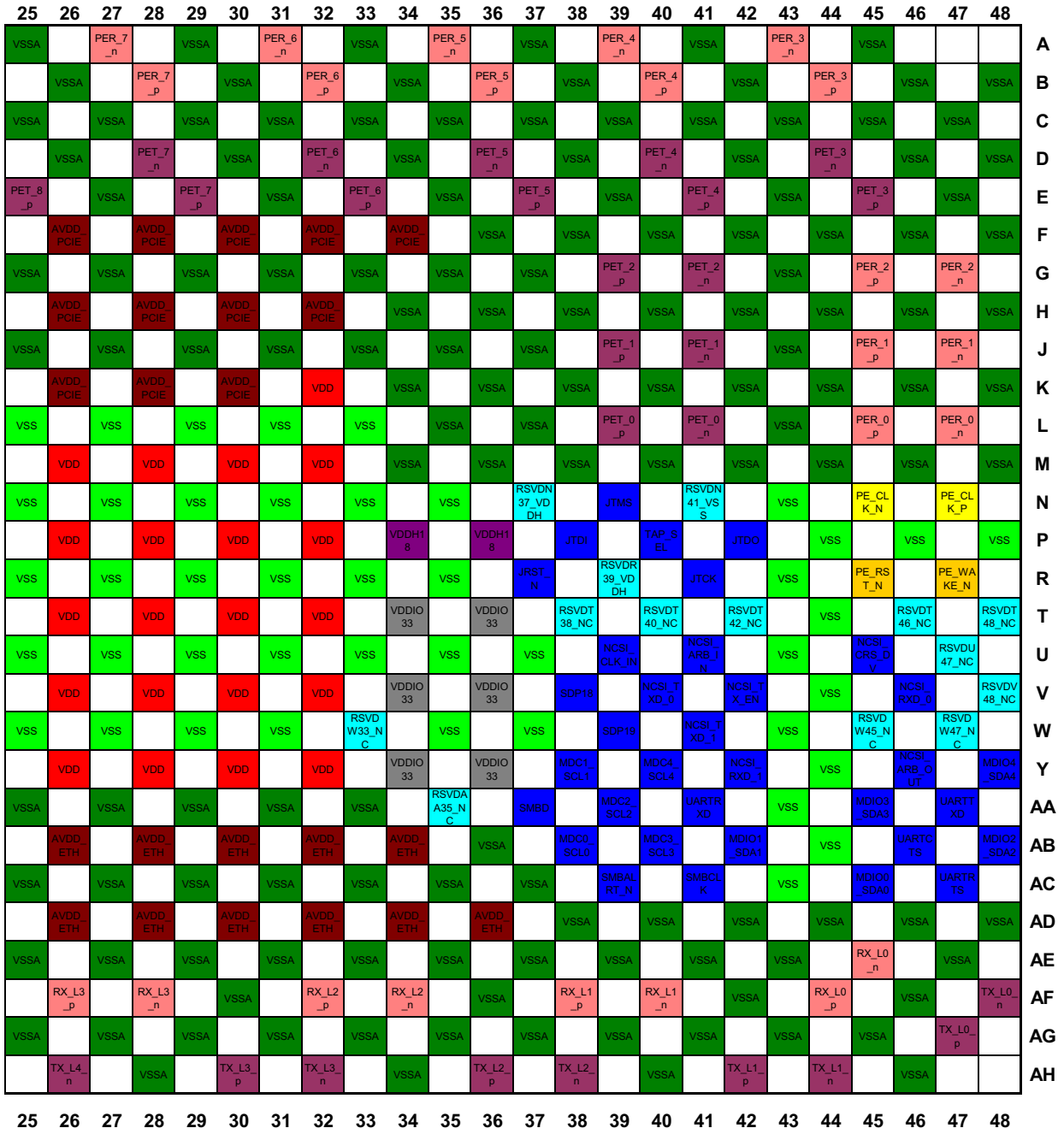
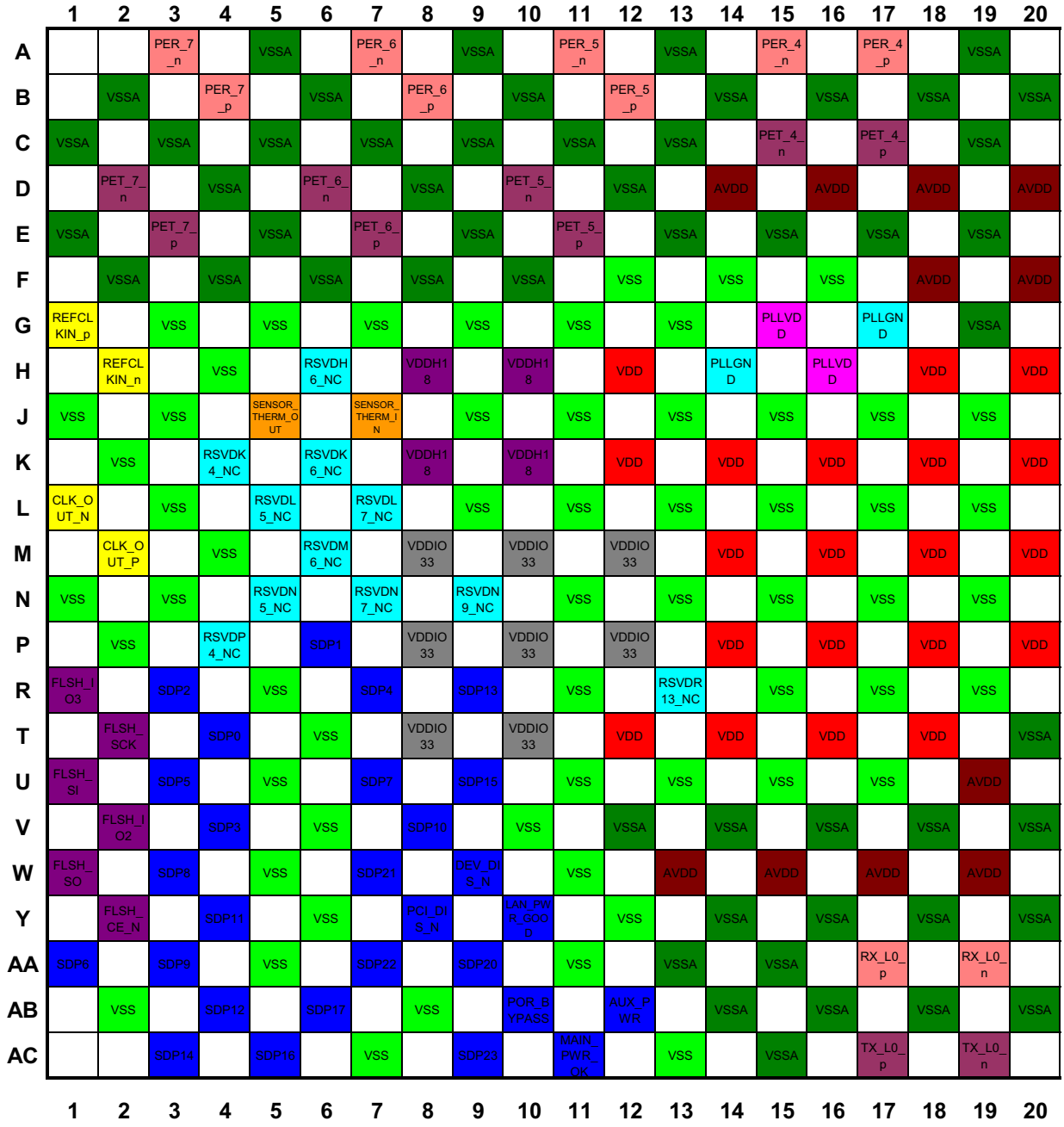


Figure 2-2. E810-CAM2/CAM1 Package Layout (Part 2)



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A			PER_7_n		VSSA		PER_6_n		VSSA		PER_5_n		VSSA		PER_4_n		PER_4_p		VSSA	
B		VSSA		PER_7_p		VSSA		PER_6_p		VSSA		PER_5_p		VSSA		VSSA		VSSA		VSSA
C	VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		VSSA		PET_4_n		PET_4_p		VSSA	
D		PET_7_n		VSSA		PET_6_n		VSSA		PET_5_n		VSSA		AVDD		AVDD		AVDD		AVDD
E	VSSA		PET_7_p		VSSA		PET_6_p		VSSA		PET_5_p		VSSA		VSSA		VSSA		VSSA	
F		VSSA		VSSA		VSSA		VSSA		VSSA		VSS		VSS		VSS		AVDD		AVDD
G	REFCLKIN_p		VSS		VSS		VSS		VSS		VSS		VSS		PLLVDD		PLLGND		VSSA	
H		REFCLKIN_n		VSS		RSVDH6_NC		VDDH18		VDDH18		VDD		PLLGND		PLLVDD		VDD		VDD
J	VSS		VSS		SENSOR_THERM_OUT		SENSOR_THERM_IN		VSS		VSS		VSS		VSS		VSS		VSS	
K		VSS		RSVDK4_NC		RSVDK6_NC		VDDH18		VDDH18		VDD		VDD		VDD		VDD		VDD
L	CLK_OUT_N		VSS		RSVDL5_NC		RSVDL7_NC		VSS		VSS		VSS		VSS		VSS		VSS	
M		CLK_OUT_P		VSS		RSVDM6_NC		VDDIO33		VDDIO33		VDDIO33		VDD		VDD		VDD		VDD
N	VSS		VSS		RSVDN5_NC		RSVDN7_NC		RSVDN9_NC		VSS		VSS		VSS		VSS		VSS	
P		VSS		RSVDP4_NC		SDP1		VDDIO33		VDDIO33		VDDIO33		VDD		VDD		VDD		VDD
R	FLSH_IO3		SDP2		VSS		SDP4		SDP13		VSS		RSVDR13_NC		VSS		VSS		VSS	
T		FLSH_SCK		SDP0		VSS		VDDIO33		VDDIO33		VDD		VDD		VDD		VDD		VSSA
U	FLSH_SI		SDP5		VSS		SDP7		SDP15		VSS		VSS		VSS		VSS		AVDD	
V		FLSH_IO2		SDP3		VSS		SDP10		VSS		VSSA		VSSA		VSSA		VSSA		VSSA
W	FLSH_SO		SDP8		VSS		SDP21		DEV_DIS_N		VSS		AVDD		AVDD		AVDD		AVDD	
Y		FLSH_CE_N		SDP11		VSS		PCI_DIS_N		LAN_PWR_GOOD		VSS		VSSA		VSSA		VSSA		VSSA
AA	SDP6		SDP9		VSS		SDP22		SDP20		VSS		VSSA		VSSA		RX_L0_p		RX_L0_n	
AB		VSS		SDP12		SDP17		VSS		POR_BYPASS		AUX_PWR		VSSA		VSSA		VSSA		VSSA
AC			SDP14		SDP16		VSS		SDP23		MAIN_PWR_OK		VSS		VSSA		TX_L0_p		TX_L0_n	

Figure 2-3. E810-XXVAM2 Package Layout (Part 1)

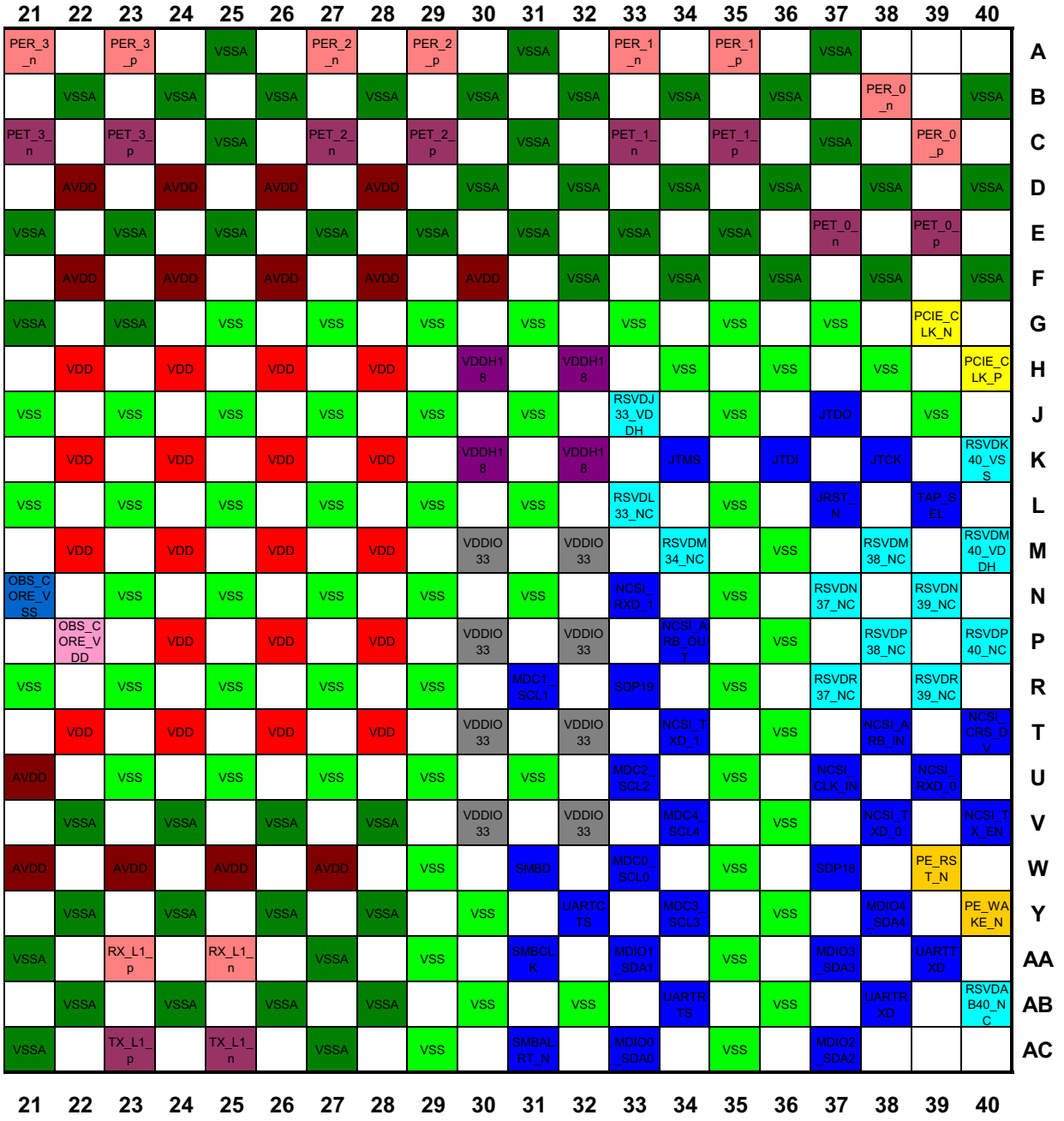


Figure 2-4. E810-XXVAM2 Package Layout (Part 2)

NOTE: ***This page intentionally left blank.***

Chapter 3 Interconnects

3.1 PCI Express (PCIe)

3.1.1 Features

The E810 supports Rev. 4.0 of the PCIe base specification.

In addition to the capabilities required by the PCIe specifications, the E810 also supports the following optional functionality as described in this section:

- All PCI functions are native PCIe functions.
- Physical Layer:
  - Support for 2.5GT/s, 5GT/s, 8GT/s, and 16GT/s
  - Interface width of 1, 4, 8, or 16 PCIe lanes
  - 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 payload size: 512B
    - Maximum read request size: 4 KB
  - Extended tags:
    - 10-bit tag as completer
    - 8-bit tag as requester
  - Function-Level Reset (FLR).
- Reliability:
  - Advanced Error Reporting (AER)
  - End-to-End CRC (ECRC) generation and checking
  - Recovery from data poisoning
  - Completion timeout
- Power Management:
  - Wake capability

- DFT and DFM support for high-volume manufacturing.
- The E810 supports the following extended capabilities:
  - AER
  - Device Serial Number (DSN)
  - Alternative RID Interpretation (ARI)
  - Single Root I/O Virtualization (SR-IOV)
  - Access Control Services (ACS)
  - TLP Processing Hints (TPH)
  - Process Address Space ID (PASID)

Table 3-1 lists the new requirements for the PCIe interface in the E810.

Table 3-1. PCIe Interface New Features

Description
General • Support PCIe base specification Ver. 4.0). • Support up to 8 PCI physical functions. ***Note:*** The PCI functions can be used in different manners per usage (for example, allocated to Ethernet ports).
PCI and PCIe Capabilities • PASID support. • Gen4 support.

3.1.2 Transaction Layer

3.1.2.1 Transactions Accepted by the E810

Table 3-2 lists the transactions accepted by the device and their attributes. See Section 3.1.2.8 for the number of credits provided per FC type.

Table 3-2. 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 messages contain payload.

Flow Control types:

- 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:

- CSR writes are 32-bit or 64-bit only.
- 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 completer abort. Data is dropped and an error is generated as per PCIe rules.

Read accesses:

- CSR reads are 32-bit or 64-bit only.

  Note: Some 64-bit reads are handled atomically, such as not interleaved with any other read requests. This applies mainly to reading counters, where all 64 bits need to be read simultaneously. Such registers are explicitly marked in their description.

- 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 register 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 as per PCIe rules.

Rules for accessing the doorbell space in the memory BAR:

Write accesses:

- Write accesses are supported within the limit of the respective PCIe credits.
- A write access that crosses over into the Flash address space is handled as a completer abort.
- 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).

Read accesses:

- Read accesses are replied with stale data. No error indication is provided.

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

Write accesses:

- Writes to Flash are silently ignored.

Read accesses:

- Reads to Flash are 32 bits wide.
- Partial reads with at least one byte disabled are handled internally as a full read. 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 completer abort. The completion includes a CA status and an error is generated as per PCIe rules.

3.1.2.3 I/O Accesses

Rules for accesses to the I/O BAR:

Write accesses:

- Write accesses are 32 bits 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 completer abort. Data is dropped and an error is generated as per PCIe rules.

Read accesses:

- Reads to the I/O BAR are 32 bits 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.
- Larger CSR read requests are handled as completer abort. The completion includes a CA status and an error is generated as per PCIe rules.
- See Section 13.1.1.5 for more details.

3.1.2.3.1 Messages

MCTP messages might contain a payload of up to 64 bytes.

3.1.2.4 Transactions Initiated by the E810

Table 3-3 lists the transactions initiated by the device and their attributes.

Table 3-3. 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 + PD

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 mode: 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 mode: *Max Payload Size* is determined solely by the setting in Function 0.
- MAX_READ_REQUEST_SIZE - The E810 supports read requests of up to 4 KB.
  - The actual maximum size of a read request is defined as the minimum {4 KB, *Max Read Request Size* field in the Device Control Register}.
- Extended tags are supported for Memory Read Requests.
- Prefixes can be added to memory read and write requests when PASID is enabled.

Note: If Extended tags are not enabled, the performance of the device might degrade. To ensure usage of extended tags, all functions must enable them in the *Extended Tag Field Enable* field.

3.1.2.4.1 Data Alignment

Requests must never specify an address/length combination that causes a memory space access to cross a 4 KB boundary. Therefore, the E810 breaks requests into 4 KB-aligned requests (if needed). This does not pose 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 general rules for packet alignment are as follows. Note that these apply to all E810 requests:

- The length of a single request does not exceed the PCIe limit of *Max Payload Size* for write, and *Max Read Request Size* for read.
- 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 meets the general rules for packet alignment, it is not broken at the cache line boundary, but rather is sent as a single packet. However, if any of the general rules require that the request be 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 = 256B 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 240 bytes + 0x10 = 256 bytes is on the cache-line boundary).

The system cache line size is controlled by the GLPCI_CNF2.*CACHELINE_SIZE* bit, loaded from the NVM *Cache Line Size* field. Note that the Cache Line Size register in the PCI configuration space is not related to the GLPCI_CNF2.*CACHELINE_SIZE* and is solely for software use.

3.1.2.5 Messages

Table 3-4 lists the response to messages sent to the device. Unlisted messages are not supported by the device and are treated as unsupported requests.

Table 3-4. Messages in the E810 (as a Receiver)

Message Code [7:0]	Routing r2r1r0	Message	E810 Response
0x00	011b	Unlock	Silently drop.
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 000b	Vendor_defined type 1	Silently drop.
0x7F	010b 011b	Vendor_defined type 1. See Section 3.1.2.5.1	Send to MCTP reassembly if Vendor ID = 0x1AB4 (DMTF) and VDM code - 0000b (MCTP). Otherwise, silently drop.

Table 3-5 lists the messages sent by the device.

Table 3-5. Messages in the E810 (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
0x1B	101b	PME_TO_Ack
0x7F	000b 010b 011b	Vendor Defined Messages (VDM); see Section 3.1.2.5.1.

3.1.2.5.1 VDM

The following vendor defined message is supported:

- DMTF
- MCTP

3.1.2.5.1.1 MCTP VDMs

MCTP VDMs are supported as both master and target. The following header fields are involved (see Section 12.7.2.1 for more details):

- Fmt — Set to 11b to indicate a 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).
- Error Present — Set to 0b.
- Attributes[1:0] — Set to 01b (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).

3.1.2.6 Transaction Attributes

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

The E810 supports only TC = 0b and VC = 0b (default).

3.1.2.6.2 TLP Processing Hints (TPH)

The E810 supports the TPH capability defined in the PCI Express specification. It does not support extended TPH requests.

Existence of a 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 E810 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 E810 exposes a PCIe TPH capability structure (see Section 14.4.5) with no steering table.

Required steps to enable TPH usage:

1. For a given function, the *TPH Requester Enable* field in the PCIe configuration TPH Requester Control register should be set to either 01b or 11b, and the *ST Mode Select* field should be set to one of the two supported values: 000b (No Table Mode) or 010b (Device Specific Mode). If this is not the case, the PF driver should not enable the TPH in the transmit and receive queue contexts.
2. Appropriate TPH *Enable* bits in the receive or transmit queue context should be set.
3. Processing hints should be programmed in the GLTPH_CTRL.*Desc_PH* and GLTPH_CTRL.*Data_PH* Processing Hints fields.
4. Steering information should be programed in the *CPUID* fields in the receive or transmit queue context.

The Processing Hints and Steering Tags are set according to the characteristics of the traffic as listed in Table 10-2.

Note: To enable TPH usage, all the memory reads are done without setting any of the byte enable bits.

3.1.2.6.2.1 Steering Tag and Processing Hint Programming

Each type of DMA traffic uses a different policy to define how the steering tag (socket ID) and processing hints are generated:

- The policy for LAN traffic is described in Section 10.3.
- The policy for offloaded traffic is described in Section 11.4.8.
- Accesses to the Host Memory Cache do not use TPH hints.
- Accesses to the admin command queues do not use TPH hints.

3.1.2.6.3 PASID Prefix

Memory read and write transactions can have a PASID prefix added if the transaction is related to a queue from a PASID-enabled VSIs and PASID is enabled in the system.

The PASID prefix is described as follows:

31 29	28	27 24	23	22	21 20	19 0
100	1	0001	E	P	Res	PASID[19:0]

where:

- The *PASID[19:0]* value is extracted from the VSI_PASID.*PASID[19:0]* field.
- E = Execute Privilege = 0
- P = Privileged Mode Requested = 0

The addition of the prefix is controlled by the VSI_PASID.*EN[31]* bit if PASID is enabled in the PASID capability. See Section 14.4.9 for more information.

3.1.2.7 Device Ordering Rules

The E810 meets the PCIe ordering rules as follows:

Deadlock avoidance — The E810 meets the PCIe ordering rules that prevent deadlocks:

- Posted writes overtake stalled read requests. This applies to both target and master directions. For example, if master read requests are stalled due to lack of credits, master 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.
- Target posted writes overtake stalled target configuration writes.
- Completions overtake stalled read requests. This applies to both target and master directions. For example, if master read requests are stalled due to lack of credits, completions generated by the E810 are allowed to proceed.

Consistency of data:

- Descriptor/Data Ordering — The E810 insures that a Rx-Descriptor is written back on PCIe only after the data that the descriptor relates to is written to the PCIe link.
- Target NP read requests might pass target posted writes addressing different PCI functions.
- Completions for target reads (memory, I/O, configuration) do not pass previous posted requests. Here are some specific usages of this rule:
  - Flush following a reset (such as FLR, BME, D3 entry, VFE clear) — When the system issues a reset event, it needs to identify when the device stops sending new posted requests from the function(s) under reset. The completion to the config write of these reset events is sent after all requests/completions related to that function(s) are flushed out. The device is expected not to issue any new posted transactions from the function(s) under reset.
  - MSI and MSI-X Ordering Rules — System software can 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.
    - Since software does not know when the tables are actually updated in the E810, a common scheme is to issue a read request to the MSI or MSI-X table after an update to the 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.
    - 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.

Independence between target and master accesses:

- The acceptance of a target posted request does not depend upon the transmission of any TLP.
- The acceptance of a target non-posted request does not depend upon the transmission of a non-posted request.
- Accepting a completion does not depend upon the transmission of any TLP.

3.1.2.7.1 Processing of Target Accesses

The E810 meets the specification requirements regarding target accesses as described in Section 3.1.2.7.

In addition, the following behaviors apply:

- Target accesses from different functions might be processed in a different order than the order in which they arrive.
- Completions that belong to requests from different PCI functions might be issued in a different order than the order of the respective requests.

3.1.2.7.2 Relaxed Ordering

The E810 takes advantage of the relaxed ordering rules in PCIe. By setting the relaxed ordering bit in the packet header, the E810 enables the system to optimize performance in the following cases:

- Relaxed ordering for LAN descriptor and data reads — When the E810 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.
  - The GLLAN_RCTL_1.*RXDESCRDROEN* bit (loaded from NVM) enables relaxed ordering for Rx-Descriptor reads.
- Relaxed ordering for LAN Rx data writes — When the E810 issues Rx 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.
  - The GLLAN_RCTL_1.*RXDATAWRROEN* bit (loaded from NVM) enables relaxed ordering for Rx data writes.
- The E810 does not relax ordering for the following requests:
  - LAN descriptor writes.
  - LAN Tx head write back.
  - Interrupt messages.
  - MCTP messages.
  - Protocol Engine traffic.
  - HMC requests.
  - EMP requests.
  - Any other requests not previously mentioned.

Relaxed ordering is globally enabled in the E810 by clearing the GLPCI_CNF2.*RO_DIS* bit, originally loaded from NVM. It is further controlled through the *Enable Relaxed Ordering* bit in the PCIe Device Control register (see Section 14.3.5.5).

3.1.2.7.3 ID-Based Ordering (IDO)

ID-based ordering was introduced in the PCIe rev. 2.1 specification. When enabled, the E810 sets IDO in all applicable TLPs defined in the PCIe specification. IDO is not set for MCTP packets.

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

- IDO is not disabled from the NVM. Device default is enabled. The value loaded from the NVM is reflected in the GLPCI_CAPSUP register.
- The PCIe *IDO Request Enable* bit (for requests) or the *IDO Completion Enable* bit (for completions) in the Device Control 2 register is set (see Section 14.3.5.11).

3.1.2.8 Flow Control

3.1.2.8.1 Flow Control Rules

The E810 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 (per Device)
Posted Request Header (PH)	Target write (one unit) Message (one unit)	96 header credit units.
Posted Request Data (PD)	Target write Message	288 data credits units.
Non-Posted Request Header (NPH)	Target read (one unit) Configuration read (one unit) Configuration write (one unit)	Four units (to enable concurrent target accesses).
Non-Posted Request Data (NPD)	Configuration write (one unit)	Four 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 E810 follows the PCIe recommendations for frequency of UpdateFC FCPs.
- Specific rules apply in L0 or L0s link state. See the PCIe specification.

3.1.2.8.2 Flow Control Timeout Mechanism

The E810 implements the optional flow control update timeout mechanism. See the PCIe specification.

3.1.2.9 End-to-End CRD (ECRC)

The E810 supports ECRC as defined in the PCIe specification. The following functionality is provided:

- Inserting ECRC in transmitted TLPs:
  - The E810 indicates support for inserting ECRC in the *ECRC Generation Capable* bit of the PCIe Configuration registers. This bit is loaded from the global *ECRC Generation Capable* NVM bit.
  - Inserting ECRC is enabled per function 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 the MCTP specification).
- 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 E810 indicates support for ECRC checking in the *ECRC Check Capable* bit of the PCIe Configuration registers. This bit is loaded from the global *ECRC Check Capable* 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

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 E810 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 E810 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 EDB Nullifying (End Bad)

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

3.1.3.4 Retry Buffer

The retry buffer size is 8 KB.

3.1.4 Physical Layer

3.1.4.1 Link Speed

The E810 supports Gen 1 (2.5GT/s), Gen 2 (5GT/s), Gen 3 (8GT/s), and Gen 4 (16GT/s).

The following configuration controls link speed:

- PCIe *Supported Link Speeds* bit — Indicates the link speeds supported by the E810.
- 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 the highest link speed supported defined by the previous *Supported Link Speeds*.

The E810 does not initiate a hardware autonomous speed change.

The E810 supports entering compliance mode at the speed indicated in the *Target Link Speed* field in the PCIe Link Control 2 register (see Section 14.3.5.13). Compliance mode functionality is controlled via the PCIe Link Control 2 register.

3.1.4.2 Link Width

The E810 supports a maximum link width of x16, x8, x4, or x1.

The maximum link width supported is loaded from the NVM into the *Maximum Link Width* field of the PCIe Link Capabilities register (see Section 14.3.5.7). Hardware default is the x16 link.

During link configuration, the platform and the E810 negotiate on a common link width. The link width must be one of the supported PCIe link widths (x1, x4, x8, x16), such that:

- If maximum link width = x16, the E810 negotiates to either x16, x8, x4, or x1.
- If maximum link width = x8, the E810 negotiates to either x8, x4, or x1.
- If maximum link width = x4, the E810 negotiates to either x4 or x1.
- If maximum link width = x1, the E810 only negotiates to x1.

The E810 does not initiate a hardware autonomous link width change.

3.1.4.3 Lane Configurations

The E810 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 E810 does not initiate lane reversal. The Link Partner (LP) might initiate lane reversal (to end up with an optimal lane width) and the E810 consents with the lane reversal.
- If lane 0 is found invalid, the E810 initiates lane reversal. Lane reversal succeeds if the LP supports link reversal.
- If the lanes at both ends of the port (such as lanes 0 and 15 for x16, lanes 0 and 7 for x8, lanes 0 and 3 for x4, lane 0 for x1) are invalid, a link is not established.

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

Figure 3-1, Figure 3-2, Figure 3-3 and Figure 3-4 depict the initial link width configuration and link degradation options. In Figure 3-2 and Figure 3-3, the upper part of the figures describe link options where the LP and the E810 are aligned. The bottom part of the figures describe link options where the LP and the E810 are reversed in order.

- Figure 3-1 applies when both the LP or the E810 is physically set to x1.
- Figure 3-2 applies when either the LP or the E810 is physically set to x4 and both are not physically set to x8.
- Figure 3-3 applies when either the LP or the E810 is physically set to x8.
- Figure 3-4 applies when either the LP or the E810 is physically set to x16.

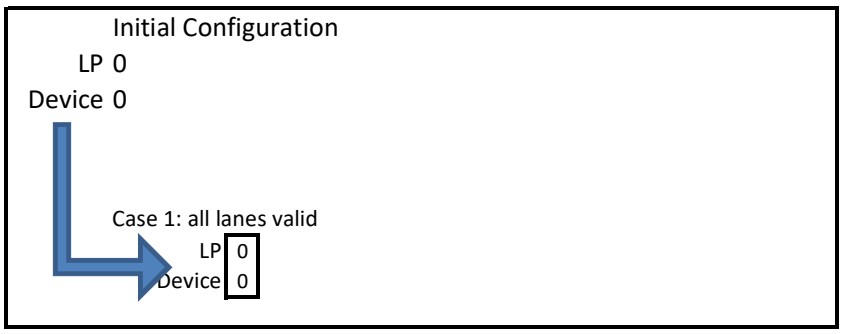


Figure 3-1. Link Width Configurations for a x1 Port

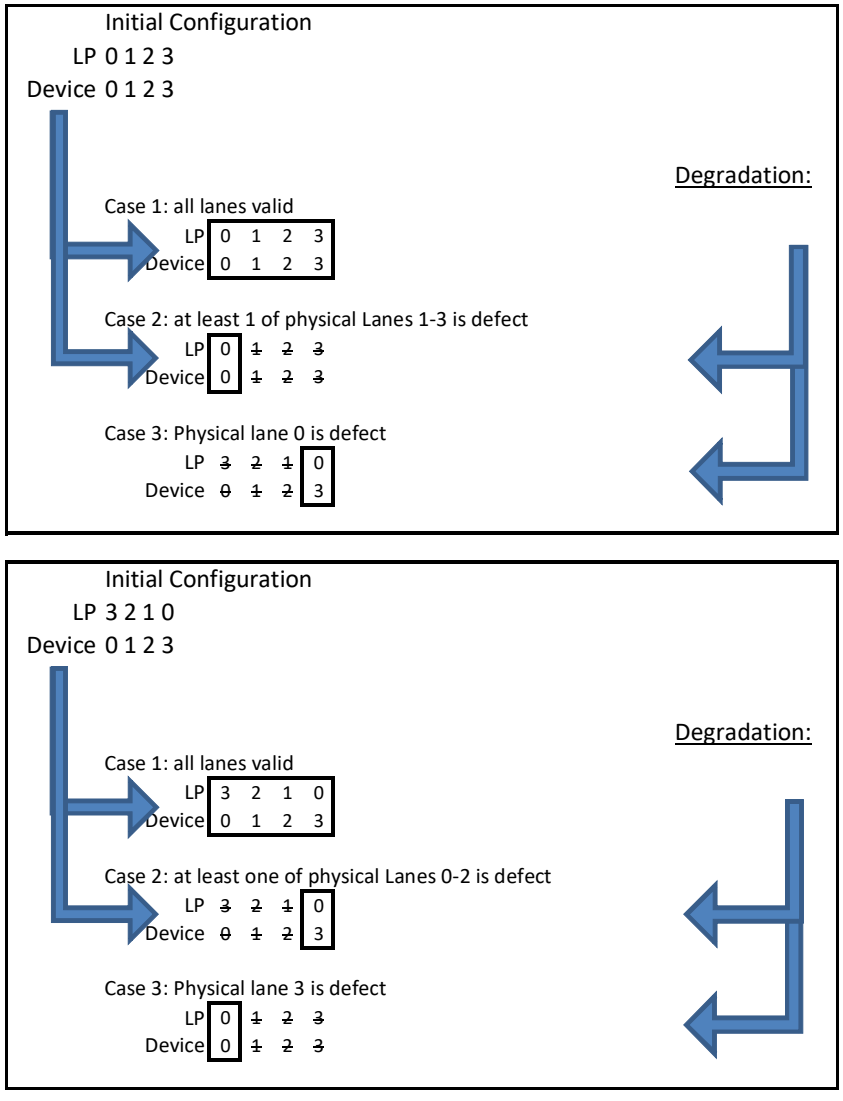


Figure 3-2. Link Width Configurations for a x4 Port

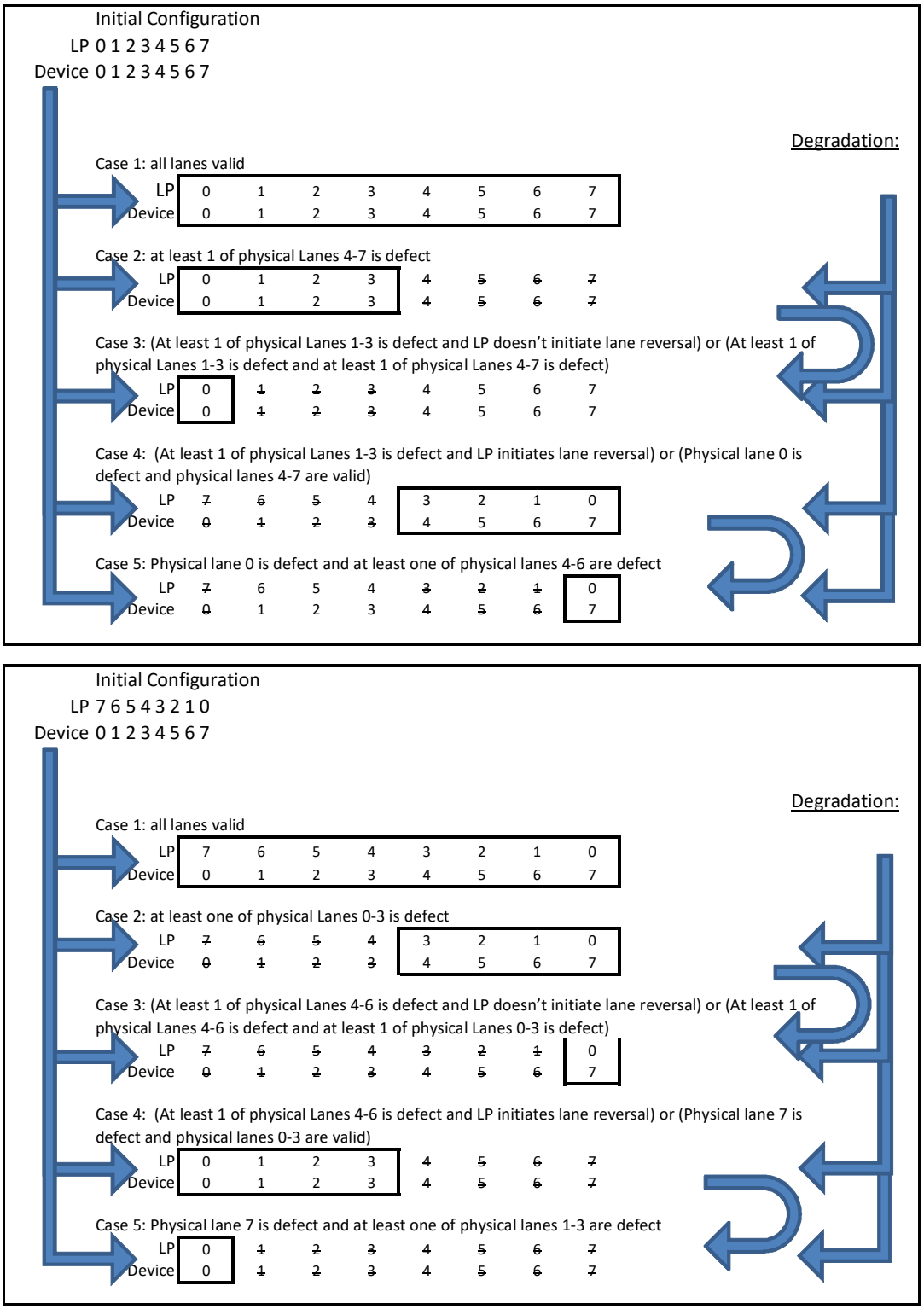


Figure 3-3. Link Width Configurations for a x8 Port

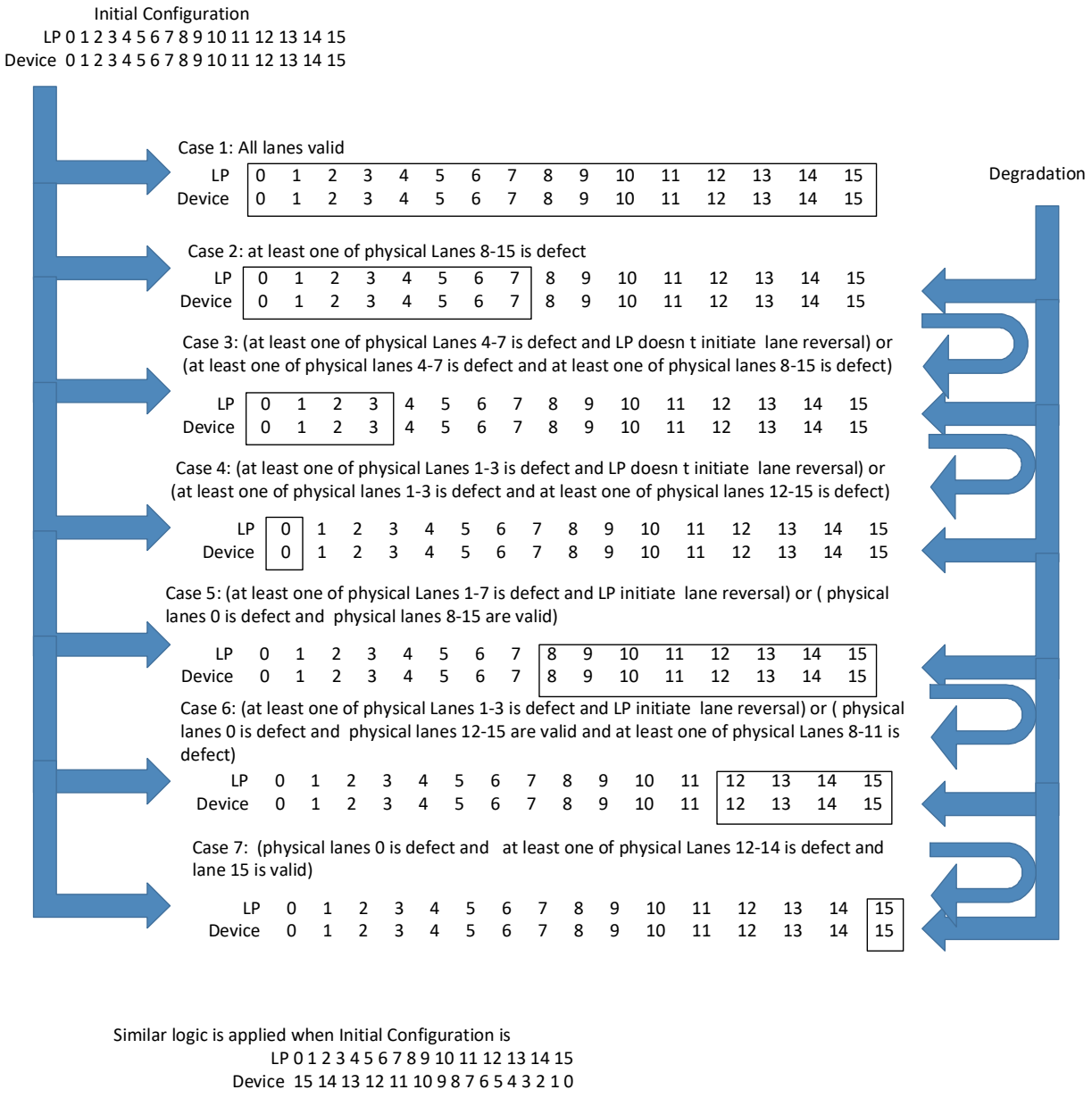


Figure 3-4. Link Width Configurations for a x16 Port

3.1.4.4 Receiver Framing Requirements

This section applies to Gen 3 or Gen 4 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=0b
- Mixed order sets across lanes (which anyway ending up with recovery)

3.1.5 Error Events and Error Reporting

This section describes error reporting and advanced error reporting.

3.1.5.1 General Description

PCIe defines two error reporting paradigms:

- Baseline capability
- Advanced Error Reporting (AER) capability.

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 E810, but the AER capability must be enabled in the NVM.

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 (see Section 6.2.4 in the *PCIe Base Specification*). Figure 3-5 shows, in detail, the flow of error reporting in PCIe. See also Figure 6-2 in the *PCIe Base Specification*.

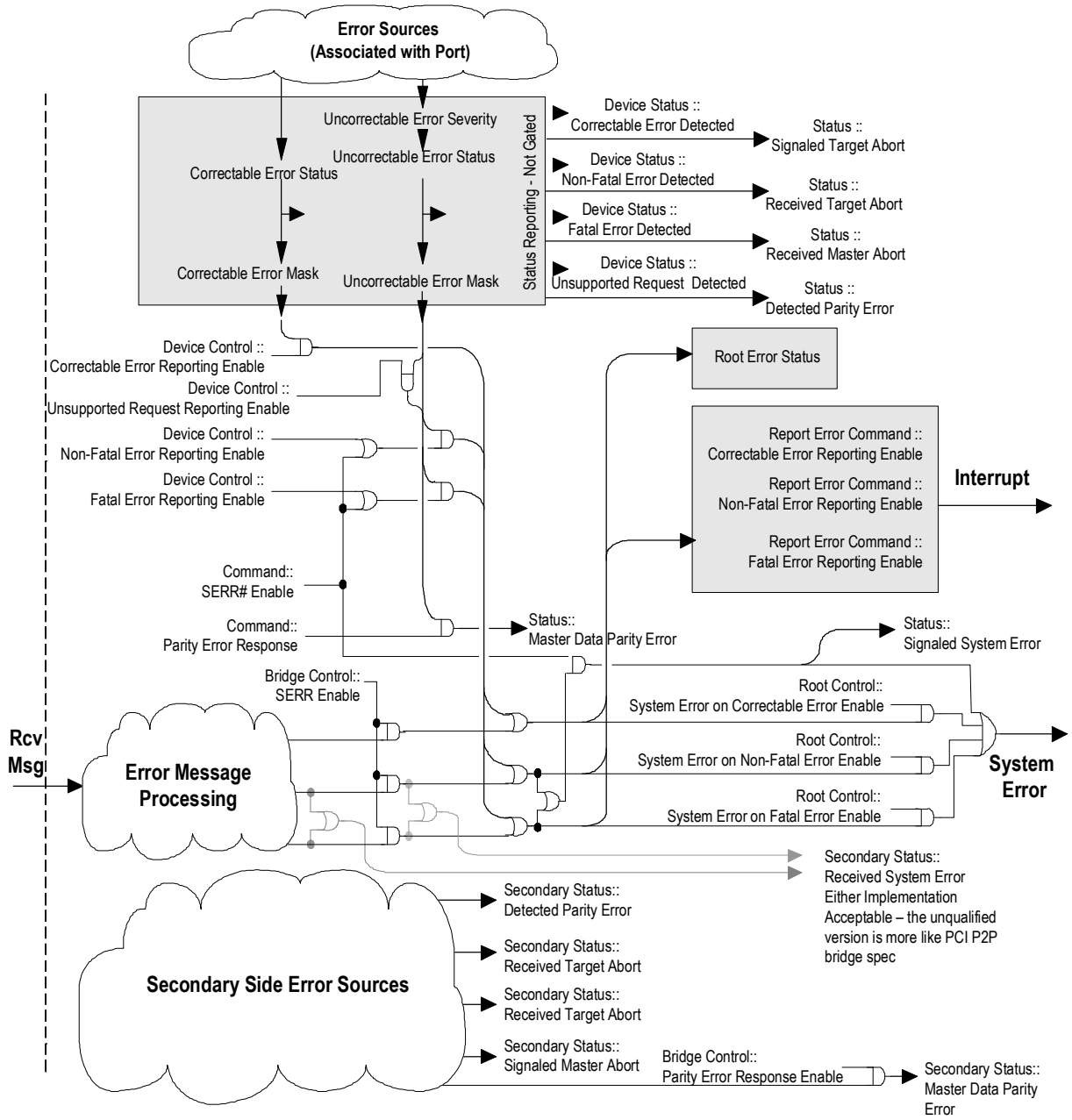


Figure 3-5. Error Reporting Mechanism

3.1.5.2 Error Events

Table 3-7 lists the error events identified by the E810 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 Not Legal 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 Protocol Error	Violations of Flow Control Initialization Protocol	Uncorrectable Send ERR_FATAL	
Poisoned TLP Received	TLP With Error Forwarding	Uncorrectable ERR_NONFATAL Log Header	See section Section 3.1.5.4 for more details. 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.
ECRC Check Failed	Failed ECRC check	Uncorrectable ERR_NONFATAL Log Header	See Section 3.1.2.9 for more details. If error is defined as non-fatal (default severity): • Send an ERR_NONFATAL message. If error is defined as fatal: • Send ERR_FATAL 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 Reception of packet with unexpected PASID prefix Received Target Access with illegal data size per Section 3.1.2.2	Uncorrectable ERR_NONFATAL Log header (including optional prefix)	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	See Section 3.1.5.3 for more details. Log the header of the Request TLP that encountered the error. 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.
Unexpected Completion	Received Completion Without a Request For It (Tag, ID, and so on)	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 Max_Payload_Size 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 (no credits for completions)

3.1.5.3 Completion Timeout Mechanism

The E810 supports completion timeout as defined in the PCIe specification.

The E810 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 14.3.5.10) is hard-wired to 1b to indicate that disabling completion timeout is supported.
  - The PCIe *Completion Timeout Disable* bit in Device Control 2 register (Section 14.3.5.11) controls whether completion timeout is enabled.
- A programmable range of timeout values.
  - The E810 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.

The following sequence takes place when completion timeout is detected:

1. The appropriate message is sent on PCIe as listed in Table 3-7.
2. 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.
3. 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 statistics 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 following sequence takes place when a poisoned TLP is received:

1. The appropriate message is sent on PCIe as listed in Table 3-7.
2. 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 inform 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, and so on.

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 GLPCI_UPADD register blocks master 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 (TLPs are sent unconditionally).

Processing a blocked transaction:

- Write transaction:
  - The transaction is dropped.
- Read transaction:
  - The transaction is dropped.
  - Set the exceeded upper address limit (read requests) event in the PCIe errors register (see Section 3.1.6).
  - The affected queue or client takes action based on the nature of the original request. An interrupt is issued to the respective PF. See Section 9.2.2.2.1.

3.1.6 Performance and Statistics Counters

The E810 incorporates counters to track the behavior and performance of the PCIe interconnect.

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).

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 E810 are listed in Table 3-8.

Table 3-8. PCIe Statistic Events Encoding

Event	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.
Poisoned TLP	0x13	Number of TLPs received with poisoned data.
ECRC Check	0x14	Number of TLPs that fail 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 three times.
Physical Layer Events		
Receive Error	0x50	Increment when one of the following occurs: • Decoder error occurred during training in the PHY. It is reported only when training ends. • 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.

3.1.6.1.1 Count Mode

The following CSR fields control operation of the Count mode:

- Four 32-bit counters GLPCI_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 GLPCI_GSCN_0_3[0] and GLPCI_GSCN_0_3[1] form the first 64-bit counter, while registers GLPCI_GSCN_0_3[2] and GLPCI_GSCN_0_3[3] form the second 64-bit counter.
    - The GLPCI_GSCL_1_P.*GIO_64_BIT_EN* selects between 32-bit and 64-bit modes.
- The GLPCI_GSCL_1_P.*GIO_COUNT_EN_[3:0]* bits enable each of the four counters.
  - The enable bits for the two 64-bit counters are GLPCI_GSCL_1_P.*GIO_COUNT_EN_0* and GLPCI_GSCL_1_P.*GIO_COUNT_EN_2*, respectively.
- The GLPCI_GSCL_1_P.*GIO_COUNT_START* bit starts event counting of enabled counters.
- The GLPCI_GSCL_1_P.*GIO_COUNT_STOP* bit stops event counting of running counters.
- The GLPCI_GSCL_1_P.*GIO_COUNT_RESET* bit resets the event counters.

- The GLPCI_GSCL_2 associates an event with each of the four 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 (GLPCI_GSCL_1_P).
- The LBC is controlled by the *GIO_COUNT_START*, *GIO_COUNT_STOP*, *GIO_COUNT_RESET* bits in the PCIe Statistic Control Register #1 (GLPCI_GSCL_1_P).
- The LBC increments every time the respective event occurs.
- The LBC is decremented every T μs as defined in the *LBC_TIMER_N* field in the PCIe Statistic Control Register #5...#8 (GLPCI_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 Register #5...#8 (GLPCI_GSCL_5_8), the respective statistics counter increments, and the LBC counter is cleared to zero.

3.1.6.2 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.

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

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

Resetting the counters configuration:

1. Set the XPPERFCON.*GRST* bit.
2. Clear the XPPERFCON.*GRST* bit (otherwise the logic stays in reset)

Setting an event:

1. Write 0x0...0 to the XPPMCL[1:0] registers.
2. Set the XPPMR[1:0].*CENS* field to 0x1.
3. Set the XPPMR[1:0].*CNTMD* field to 0x1.
4. Set the XPPMER[1:0].*XPRSCA* field to 0x1.
5. Set the event according to Table 3-9.

Starting a count:

1. Set the XPPERFCON.*GCE* bit

Stopping a count:

1. Clear the XPPERFCON.*GCE* bit

Reading the count (note: reading the counter clears their values):

1. Read the respective XPPMDH[1:0] and XPPMDL[1:0] register pair in a single 64-bit aligned access.

Table 3-9 defines the Link and Physical Layer events:

Table 3-9. 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].*TXL0SU*
Rx L0s State Utilization	Counts the number of entries to L0s on the Rx lanes.	XPPMER[1:0].*RXL0SU*
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 (that is, 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].*RXL0STXL0SU*
NAK DLLP Received	Counts number of received NAK DLLPs.	XPPMER[1:0].*NAKDLLP*

3.1.6.2.1 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:

1. Set the XPPMR[1:0].*CENS* field to 0x1.
2. Set the XPPMR[1:0].*EGS* field to 0x2.
3. Set the XPPMER[1:0].*FCCSEL* field to the desired traffic type (posted, non-posted, completions, or all).
4. Set the XPPMER[1:0].*TXRXSEL* field to desired values.
5. Set the XPPMER[1:0].*XPRSCA* field to 0x1.
6. Set the XPPERFCON.*GCE* field to 0x1.

Registers fields used exclusively by the bandwidth counters:

- XPPMER[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.

3.2 Ethernet Interconnect

3.2.1 Media Access Control (MAC) Layer

The E810 supports up to eight full-duplex Ethernet MAC ports compliant with IEEE Std802.3-2018 standard (Clause 4 and Annex 4A). The MAC ports can be configured to operate at different speeds, as explained in Section 3.2.1.2.

Each MAC port is associated with a corresponding Media Access Unit (MAU) that provides the physical layer interfaces. The MAUs must be configured to operate with the appropriate physical layer protocols based on the MAC operating speed. Physical layers supported by the E810 for different speeds of operation are explained in Section 3.2.2, "Physical Layer Interface".

3.2.1.1 MAC Features

Table 3-10 lists the E810 Ethernet port features.

Table 3-10. Link Layer Ethernet Port Features

Description
Ethernet Speed and Interfaces: • 100 Gb/s: 2 ports of CGMII • 50 Gb/s: 2 ports of XLGMII • 25 Gb/s: 4 ports of XGMII (boosted to 25G) • 10 Gb/s: 8 ports over XGMII (supporting 10G,1G, 100M) (ONPI 32 bits - reduced clock)
The E810's Maximum Transmit Unit Size (MTU) is 9728 - Ethernet header/CRC = 9728 - 18 = 9710 bytes (jumbo frames). MTU can be further reduced by additional header fields such as Virtual Local Area Network (VLAN) tag(s), and so on.
Full-duplex operation at all supported speeds.
Integrates support for IEEE Std 802.3 Clause 73 Auto-Negotiation for Backplane Ethernet (The auto-negotiation protocol is done by the PHY).

3.2.1.2 MAC Speed Configuration

Table 3-11 lists the possible speed configurations available on each MAC port.

Table 3-11. MAC Port and Possible Speed Configurations

Port	100 Mb/s	1 Gb/s	10 Gb/s	25 Gb/s	50 Gb/s	100 Gb/s
MAC 0	Y	Y	Y	Y	Y	Y
MAC 1	Y	Y	Y	Y	Y	Y
MAC 2	Y	Y	Y	Y	N	N
MAC 3	Y	Y	Y	Y	N	N
MAC 4	Y	Y	Y	N	N	N
MAC 5	Y	Y	Y	N	N	N
MAC 6	Y	Y	Y	N	N	N
MAC 7	Y	Y	Y	N	N	N

When LAN ports are disabled in multi-port system configurations, corresponding PCIe functions must be disabled through the NVM or external disable pins. See Section 4.4 for details on PCI function disable and LAN port disable functionality.

The link speed for each port can be controlled through NVM loaded settings or Link Configuration admin commands (see Section 3.2.4). In some physical interfaces, the final link speed is selected out of the pre-configured options, based on the link auto-negotiation protocol.

The maximum speed enabled depends on the number of enabled ports:

- 2 Ports setting:
  - Each port can be independently configured to 100/50/25/10/1 Gb, 100 Mb, or disabled.
  - In this setting, only Port0 and Port1 can be enabled.
- 4 Ports setting:
  - Each Port can be independently configured to 25/10/1 Gb, 100 Mb, or disabled.
  - In this setting, only Port0, Port1, Port2, Port3 can be enabled.
- 8 Ports setting:
  - Each Port can be independently configured to 10/1 Gb, 100 Mb, or disabled.
  - Some ports can be configured to 25 Gb, depending on other port's configurations (see Table 3-12).
  - In this setting, all ports can be enabled.

Note: The ports maximal speed depends also on the maximal bandwidth SKU. The above speeds assumes 200G SKU.

Note: The number of ports and their speed affect the E810 pipe settings, such as TPB, RPB, DCB, and more. This is done using the adaptive NVM method, when the link topology is configured. See Section 3.3.3.3.

Table 3-12 illustrates the allowed port configurations when the E810 bandwidth is not limited by SKU setting (like 200G max bandwidth SKU).

Table 3-12. Port Configurations

Ports[1]/ Configuration	0	1	2	3	4	5	6	7	
2 ports	100G	100G	-	-	-	-	-	-	2 ports (100G and below)
4 ports	25G	25G	25G	25G	-	-	-	-	4 ports (25G and below)
6 ports - 1	25G	25G	10G	10G	-	-	10G	10G	2x25G + 4x10G
6 ports - 2	10G	10G	25G	25G	10G	10G	-	-	2x25G + 4x10G
6 ports - 3	25G	10G	25G	10G	-	10G	-	10G	2x25G + 4x10G
6 ports - 4	10G	25G	10G	25G	10G	-	10G	-	2x25G + 4x10G
6 ports - 5	25G	10G	10G	25G	-	10G	10G	-	2x25G + 4x10G
6 ports - 6	10G	25G	25G	10G	10G	-	-	10G	2x25G + 4x10G
7 ports - 1	25G	10G	10G	10G	-	10G	10G	10G	1x25G + 6x10G
7 ports - 2	10G	25G	10G	10G	10G	-	10G	10G	1x25G + 6x10G
7 ports - 3	10G	10G	25G	10G	10G	10G	-	10G	1x25G + 6x10G

Table 3-12. Port Configurations [continued]

Ports[1]/ Configuration	0	1	2	3	4	5	6	7	
7 ports - 4	10G	10G	10G	25G	10G	10G	10G	-	1x25G + 6x10G
8 ports	10G	10G	10G	10G	10G	10G	10G	10G	8 ports (10G and below)

1. Port numbers in this table refer to E810 port numbering. See Section 3.2.2.3 for port mapping.

3.2.1.3 Transmit Padding

The minimum frame size for Ethernet as specified by IEEE Std 802.3 standard is 64 bytes. The E810 MAC pads, with zeros, Ethernet packets that are smaller than 64 bytes during transmit. In the E810, transmit packets arrive already padded to the MAC. Refer to the padding rules for transmitted and received packets and loop-back packets in Section 7.12.2.

3.2.1.4 Jumbo Frame Support

The E810 MAC supports transmission and reception of frames of up to 9.5 KB (9728 bytes). Maximum receive and transmit frame size is configured through the *Max Frame Size* field in the Set MAC Config command (see Section 3.2.4.1.2).

3.2.1.5 Ethernet Flow Control (FC)

The E810 supports flow control (pause) as defined in 802.3x (IEEE Std 802.3-2008 Annex 31B), as well as the specific operation of asymmetrical flow control (asymmetric pause) defined by 802.3z (IEEE Std 802.3-2008 Annex 28B). The E810 also supports Priority Flow Control (PFC) as defined in IEEE P802.1Qbb, sometimes referred to as Class Based Flow Control or (CBFC), as part of the DCB architecture.

Note: An E810 port can either be configured to receive 802.3x Link Flow Control (LFC) packets or 802.1Qbb/802.3bd PFC packets. It does not support the reception of both types of packets simultaneously over the same port.

Flow control is implemented to reduce receive buffer overflows, which result in the dropping of received packets. Flow control 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 flow control allows for one link partner to send flow control packets while being allowed to ignore their reception (for example, not required to respond to PAUSE frames).

The following registers define the basic control functionality. Refer to the registers specified in Section 8.2.4.4 for the other programming related to flow control. In Data Center Bridge (DCB) mode, some of the registers are duplicated per Traffic Class (TC), up to eight duplicate copies of the registers. If DCB is disabled, index [0] of each register is used.

3.2.1.5.1 MAC Control Frames and Reception of Flow Control Frames

3.2.1.5.1.1 MAC Control Frame — Other Than FC

IEEE 802 reserved the EtherType value of 0x8808 for MAC control frames as listed in Table 3-13. The MAC control frame format is specified in IEEE 802.3 Clause 31.

Table 3-13. MAC Control Frame Format

Field	Description
DA	The *Destination Address* field can be an individual or multicast (including broadcast) address. Permitted values for the *Destination Address* field can be specified separately for a specific control opcode such as FC packets.
SA	Port Ethernet MAC Address (6 bytes).
Type	0x8808 (2 bytes).
Opcode	The MAC control opcode indicates the MAC control function.
Parameters	The *MAC Control Parameters* 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 *Reserved* field is used when the MAC control parameters do not fill the fixed length MAC control frame.
CRC	4 bytes.

3.2.1.5.1.1.1 MAC Control Frame Receive Identification

Packets with:

- MAC DA = 01-80-C2-00-00-01.
- EtherType value of 0x8808 are considered to be control frames.

3.2.1.5.1.2 Structure of 802.3x FC Packets

802.3x FC packets are defined by the following three fields (see Table 3-14):

- A match on the 6-byte multicast address for MAC control frames, or a match to the station address of the device. The 802.3x standard defines the MAC control frame multicast address as 01-80-C2-00-00-01.
- A match on the *Type* field. The *Type* field in the FC packet is compared against an IEEE reserved value of 0x8808.
- A match of the MAC control *Opcode* field has a value of 0x0001.

Frame-based flow control 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 (slot time). A pause quanta lasts 64 byte times, which is converted in to an absolute time duration according to the line speed.

Note: XON frame signals the cancellation of the pause that was initiated by an XOFF frame. For example, a pause for zero pause quanta.

Table 3-14. 802.3x Link Flow Control - Frame Formats

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.

3.2.1.5.1.2.1 802.3x Frame Receive Identification

Received frames that are identified as control frames (see Section 3.2.1.5.1.1.1) can further be classified as 802.3x if the frames opcode = 0x0001 as listed in Table 3-14.

3.2.1.5.1.3 Priority Flow Control (PFC)

EEDC introduces support for multiple TCs assigning different priorities and bandwidth per TC. LFC stops all the TCs. PFC specified in IEEE P802.1Qbb enables more granular flow control on the Ethernet link in an EEDC environment, as opposed to the PAUSE mechanism defined in 802.3x. The PFC frame format is specified in IEEE P802.3bd.

Table 3-15. Priority Flow Control - Frame Format

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 3-16. 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		

Each of the eight timers refers to a specific User Priority (UP). For example, Timer 0 refers to UP 0, and so on. The E810 binds a UP (and therefore the timer) to one of its TCs according to the UP-to-TC binding tables. Refer to the PRTDCB_TUP2TC register (Section 13.2.2.17.101) for the binding of received PFC frames to Tx TCs, and to the PRTDCB_RUP2TC register (Section 13.2.2.17.99) for the binding of transmitted PFC frames to Rx TCs.

When a PFC frame is formatted by the E810, the same values are replicated into every *Timer* field and *Priority Enable Vector* bit of all the UPs bound to the concerned TC. These values as configured in the PRTDCB_RUP2TC register.

The following rule is applicable for the case of multiple UPs that share the same TC as configured in the PRTDCB_TUP2TC register. When PFC frames are received with different timer values for the previous UPs, the traffic on the associated TC must be paused by the highest XOFF timer's value.

3.2.1.5.1.3.1 PFC Frame Receive Identification

Received frames that are identified as control frames (see Section 3.2.1.5.1.1.1) can further be classified as PFC if the frames opcode = 0x0101 as listed in Table 3-15.

3.2.1.5.1.4 Operation and Rules

The E810 operates in either LFC mode or in PFC mode. Enabling both modes concurrently is not allowed.

Note: LFC capability must be negotiated between link partners via the auto-negotiation process. PFC capability is negotiated via some higher level protocol (DCBX), and the resolution is usually provided to the software device driver by the EEDC management agent. It is the responsibility of the EMP to reconfigure the LFC settings after the auto-negotiation process is resolved.

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

1. Increment the appropriate statistics register(s).
2. Initialize the pause timer based on the packet's PAUSE *Timer* field (overwriting any current timer's value).
   - In case of 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.
3. Disable packet transmission or schedule the disabling of transmission after the current packet completes.
   - In case of PFC, this is done per paused TC.

Resumption of transmission can occur under the following conditions:

- Expiration of the PAUSE timer.
  — In case of PFC, this is done per TC.

- Reception of an XON frame (a frame with its PAUSE timer set to 0b).
  - In case of PFC, this is done per TC.

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

3.2.1.5.1.5 Timing Considerations

When operating at 100 Gb/s line speed, the E810 does not begin to transmit a (new) frame more than 394 pause_quantum after the reception of a valid XOFF frame that contains a non-zero value of pause_time, as measured at the wires (a pause quantum is 512 bit times).

When operating at 50 Gb/s line speed, the E810 does not begin to transmit a (new) frame more than 394 pause_quantum after the reception of a valid XOFF frame that contains a non-zero value of pause_time, as measured at the wires (a pause quantum is 512 bit times).

When operating at 25 Gb/s line speed, the E810 does not begin to transmit a (new) frame more than 80 pause_quantum after the reception of a valid XOFF frame that contains a non-zero value of pause_time, as measured at the wires (a pause quantum is 512 bit times).

When operating at 10 Gb/s line speed, the E810 must not begin to transmit a (new) frame more than 60 pause quanta after receiving a valid XOFF frame, as measured at the wires. When connected to an external 10GBASE-KR PHY with FEC, or to an external 10GBASE-T PHY, the response time requirement increases to 74 pause quanta because of extra delays consumed by these external PHYs.

When operating at 1 Gb/s line speed, the E810 must not begin to transmit a (new) frame more than two pause quanta after receiving a valid XOFF frame, as measured at the wires.

The IEEE P802.1Qbb draft 2.3, specifies that the tolerated response time for priority XOFF frames is 614.4 ns (equivalent of 12 pause quanta at the link speed of 10 Gb/s). This extra budget in addition to the link delay is aimed to compensate for the fact that the decision to stop new transmissions from a specific TC must be taken earlier in the transmit data path than for the LFC case.

3.2.1.5.2 Transmitting PAUSE Frames

The E810 generates PAUSE packets to insure there is enough space in its receive packet buffers to avoid packet drop. The E810 monitors the fullness of its receive FIFOs and compares it with the contents of a programmable threshold. When the threshold is reached, the E810 sends a PAUSE frame.

The E810 supports both LFC and PFC, but not both concurrently (at the same physical port). When DCB is enabled, it sends only PFC. When DCB is disabled, it sends only LFC.

Note: Similar to the reception of flow control packets previously mentioned, software can enable flow control transmission only after it is negotiated between the link partners (possibly by auto-negotiation).

3.2.1.5.2.1 PFC

Like Rx flow control, Tx flow control operates in either a link 802.3x compliant mode or in PFC mode, but not in both at the same time.

The same flow control mechanism is used for PFC and for 802.3x flow control 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 3.2.1.5.1.2.1.

A specific consideration 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 PRTDCB_RUP2TC register).

3.2.1.5.2.2 Operation and Rules

At 10 Gb/s and lower speeds, the *TFCE* field in the Flow Control Configuration (PRTDCB_FCCFG) register enables transmission of PAUSE packets as well as selects between the LFC mode and the PFC mode.

Refer to Section 8.2.1.4.1 for the criteria used by the device for sending a XOFF frame to the neighbor. The E810 sends an additional 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.

From the time it has issued a PAUSE frame to the neighbor, the E810 starts counting down in an internal shadow counter used to mirror the pause timeout counter at the partner's end. When this internal counter reaches the value set, then, if the PAUSE condition is still valid (meaning that the buffer(s) fullness is still above the relevant watermarks), an XOFF message is sent again.

Once the receive buffer fullness reaches the relevant low watermarks, the E810 sends an XON message (a PAUSE frame with a timer value of zero). Refer to Section 8.2.1.4.1 for the criteria used by the device for sending a XON frame to the neighbor.

3.2.1.6 Inter Packet Gap (IPG) Control and Pacing

The E810 can limit the packet transmission using one of the following:

- **Programmable fixed IPG extension, per port configuration** — The inter packet gap can be extended by fixed number of 16-bytes words, for all Tx packets. This configuration can enable connection through Bump On the Wire devices that need larger inter packet gap. The *TX_IPG* field of the PRTMAC_TX_PACE[PRT] register specifies the number of 16-bytes words inserted between each consecutive packets.
- **Average data rate limitation, per port configuration** — The gap between packets can be extended in a way that the average data rate is lower than the link speed. This configuration might be used with PHYs that implements line speeds that can not be directly configured to the E810 (for example, 5 Gb/s), but can connect to the E810 with and interface with which the E810 can support its speed (for example, 10 Gb/s). Pacing rate is implemented by extending the IPG. The *TX_PACE* field of register PRTMAC_TX_PACE[PRT] specifies the number of bits added to IPG for each 16 bytes that are transmitted. The total number of bits to be added is accumulated, and when the IPG is inserted between packets, the IPG is extended in 16-bytes granularity, according to the total number of accumulated bits, while the residual is kept to be added at the next gap between packets. To meet the IEEE 802.3 specifications, 12 bytes of minimal inter frame gap are always added to the IPG.

The following examples show how the *TX_PACE* field can be calculated to achieve specific rate.

Example 1:

Assume all transmitted packets are in size of 64 bytes. To achieve a 5 Gb/s data rate when link rate is 10 Gb/s (50%) and packet length is 64 bytes (4 x 16-bytes words), programmers need to add an additional IPG, calculated as follows. The 64-byte packet is transmitted with 12 bytes IPG, and the bandwidth must be reduced by 50%, or add 1 byte for each transmitted byte or IPG byte. Therefore, for each packet 76 bytes must be added. As TX_PACE is added in bit resolution for each of the 4 16-byte words of the packet, the programmer therefore should configure 76*8/4=152 bits in the *TX_PACE* field. The multiplication by 8 is to translate bytes to bits and the division is to calculate the number of bits to be added per 16-byte word.

Example 2:

Now assume all transmitted packets are in size of 65 bytes. To achieve a 5 Gb/s data rate when link rate is 10 Gb/s and packet length is 65 bytes (5 x 16-byte words when rounded up) programmers must add an additional IPG of 65+12=77 bytes. Note that in these case, where the packet length counted in 16-byte words is not an integer, the *TX_PACE* would be added also for a fraction of 16-bytes words transmitted. As *TX_PACE* is added in bit resolution for each of the five 16-byte words of the packet, the programmer therefore should configure 77*8/5=123.2 in the *TX_PACE* field. *TX_PACE* should be rounded up to 124, to ensure transmission at a slightly lower speed, as a slightly higher speed might result in packet loss in some extreme case.

Note: For the two examples above, a fixed packet size is assumed. For the case of variable packet size, if transmission must be ensured in lower than the configured rate, choose the highest *TX_PACE*, as calculated for all packet sizes. Table 3-17 shows the exact number of bits (rounded up to integer value) that needs to be added in order to reach 50% of the line rate.

Table 3-17. Example of TX_IPG Configuration to Achieve 50% of the Data Rate for Different Packet Sizes

Packet Size	Added Bits per 16-Byte Word of the Packet (TX_IPG)
64	152
65	124
66	125
67	127
68	128
69	130
70	132
71	133
72	135
73	136
74	138
75	140
76	141
77	143
78	144
79	146
80	148
81	124
...	
1500	129

Assuming rate must be equal or less than 50% (for example, pacing to 5 Gb/s with line speed of 10 Gb/s), *TX_IPG* should be configured to 152 bits.

Table 3-18 specifies the *TX_IPG* configuration for different pacing speeds with average target rate in 10% steps, assuming evenly-distributed packet size.

Table 3-18. Configurations for Different Pacing Speeds

Pacing Speed (Percentage of Line Rate)	TX_IPG	Pacing Config Field (Bits 6:3) in the Set MAC Config Admin Command
100%	0	0000b
90%	17	1001b
80%	38	1000b
70%	66	0111b
60%	102	0110b
50%	152	0101b
40%	228	0100b
30%	355	0011b
20%	608	0010b
10%	1368	0001b
25%	456	1110b
93% (WAN)	12	1111b

3.2.1.7 MAC Speed Change at Different Power Modes

Auto-negotiation enables establishment of link speed at the Highest Common Denominator (HCD). During certain low power modes, power saving might be more important than link performance. The E810 supports an additional mode of operation, where it can configure the PHY to fixed speed, starting from the lowest speed and checking if the link can be up. This method sets the link speed to the Lowest Common Denominator (LCD) link speed. The link-up process enables the link to come up at any possible speed in cases where power is more important than performance. See further information in Section 5.2.2.

3.2.1.8 MAC Errors

The E810's MAC supports identification of the following erroneous packets:

- **L2 CRC Error** — Packet's FCS check resulted in an error.
- **Undersized or Oversized Packets** — Received frame size is smaller than 64 bytes or larger than the configured max frame size, which was either loaded from the NVM or set using the Set MAC Config command (see Section 3.2.4.1.2).
- **Illegal Byte Error** — Indication that an illegal control byte was received on the interface. An illegal control byte is any value that is not legal. (See IEEE802.3 for legal symbols, such as /I/,/E/,/T/, /D/ , /S/, /Seq/, or /LPI/).
- **Error Byte Error** — Indication that a packet was received with the error control byte (/E/) on the XGMII interface. Indicates that the PCS has encountered an error during the packet's reception.
- **802.3 Length Error** — <Length> field information in the 802.3 header does not match the packet's actual size.

Packets containing any one of the previous errors can be filtered by the device or forwarded to a pre-configured VSI based on the *SBP* flag in the PRT_SBPVSI register.

Note: Packets shorter than 64 bytes are always filtered by the MAC layer and are not affected by the Store Bad Packets (SBP) configuration.

3.2.1.8.1 MAC Error Counters

Table 3-19 lists the different MAC error counters supported by the device.

Table 3-19. MAC Error Counters

Field	Description
CRC Error	Counts the number of packets received with CRC errors that are not fragments.
Illegal Bye Error	Counts the number of packets received with an illegal control byte on the CGMII/XLGMII interface.
Error Byte	Counts the number of packets that were received with an error control byte on the CGMII/XLGMII interface.
Receive Length Error	Counts the number of packets received with error in length field comparison in the 802.3 header.
Receive Undersize	Counts the number of packets with good CRC that are smaller than 64 bytes.
Receive Fragment	Counts the number of packets with bad CRC that are smaller than 64 bytes
Receive Oversize	Counts the number of packet that are larger than the configured max frame size.
MAC Short Packet Discard	Counts the number of packets smaller than 32 bytes that were discarded by the MAC layer.

3.2.2 Physical Layer Interface

3.2.2.1 Introduction

The E810 provides up to eight Ethernet port interfaces. The following sections describe the physical interfaces that the E810 can support and manage through firmware.

Table 3-20. Supported Electrical Modes

Speed	Mode	Spec	Comments
100G	CAUI-4	• IEEE 802.3 Annex 83D/83E	• 4x25G AUI used for chip-to-chip or chip-to-module connectivity. Annex 83D/E provides electrical specs. Clause 82 PCS, Clause 91 RS(528,514) required for support of 100GBASE-SR4, CWMD4, or PSM4 optics.
	100GBASE-KR4	PMD: • IEEE 802.3 Clause 93 PCS: • IEEE 802.3 Clause 82 • IEEE 802.3 Clause 91	• 4x25G used for backplane connectivity. • Uses RS(528,514).
	100GBASE-CR4	PMD: • IEEE 802.3 Clause 92 PCS: • IEEE 802.3 Clause 82 • IEEE 802.3 Clause 91	• 4x25G used for connecting to DA Copper cable. • Uses RS(528,514).
	100GBASE-CR2	PMD: • IEEE 802.3 Clause 136 PCS: • IEEE 802.3 Clause 82 • IEEE 802.3 Clause 91	• 2x50G lanes using PAM4 modulation. Used for connecting to DA Copper cable. • Uses RS(544,514).
	100GBASE-KR2	PMD: • IEEE 802.3 Clause 137 PCS: • IEEE 802.3 Clause 82 • IEEE 802.3 Clause 91	• 2x50G lanes using PAM4 modulation. Used for backplane connectivity. • Uses RS(544,514)
	100GAUI-2	IEEE 802.3 Annex 135F/135G	• 2x50G lane AUI using PAM4 modulation and RS(544,514) for chip-to-chip or chip-to-module connectivity.
	100GAUI-4	IEEE 802.3 Annex 135D/135E	• 4x25G lane AUI using NRZ modulation and RS(544,514) for chip-to-chip or chip-to-module connectivity.

Table 3-20. Supported Electrical Modes [continued]

Speed	Mode	Spec	Comments
50G	50GBASE-KR2	PMD: • IEEE 802.3 Clause 93 like PCS: • IEEE 802.3 Clause 82 like	• Used for Backplane connectivity. • Not specified in IEEE 802.3, KR PMD with 2x25G lanes as specified in 25/50G Ethernet consortium.
	50GBASE-CR2	PMD: • IEEE 802.3 Clause 92 like PCS: • IEEE 802.3 Clause 82 like	• Used for connecting to DA Copper cable. • Not specified in IEEE 802.3, CR PMD with 2x25G lanes as specified in 25/50G Ethernet consortium.
	50GBASE-CR	PMD: • IEEE 802.3 Clause 136 PCS: • IEEE 802.3 Clause 133 • IEEE 802.3 Clause 134 RS-FEC	• 1x50G PAM4 used for connecting to DA Copper cable. • RS(544,514).
	50GBASE-KR	PMD: • IEEE 802.3 Clause 137 PCS: • IEEE 802.3 Clause 133 • IEEE 802.3 Clause 134 RS-FEC	• 1x50G PAM4 used for backplane connectivity. • RS(544,514)
	50GAUI-2	IEEE 802.3 Annex 135D/135E	• 2x25G lane AUI using NRZ modulation and RS(544,514) for chip-to-chip or chip-to-module connectivity.
	50GAUI-1	IEEE 802.3 Annex 135F/135GE	• 1x50G lane AUI using PAM4 modulation and RS(544,514) for chip-to-chip or chip-to-module connectivity.
	LAUI-2	IEEE 802.3 Annex 135B/135C	• 2x25G lane AUI using NRZ modulation and RS(528,514) or no FEC for chip-to-chip or chip-to-module connectivity.
25G	25GAUI	PMD: • 25GAUI C2C Annex 109A • 25GAUI C2M Annex 109B PCS: • IEEE 802.3 Clause 107 PCS, Clause 108 RS-FEC	• 109A used for chip-to-chip connectivity. • 109B used for connecting to optics (generally 25GBASE-SR/LR in SFP28 module with RS(544,514).
	25GBASE-KR	PMD: • IEEE 802.3 Clause 111 PCS • IEEE 802.3 Clause 107 PCS, Clause 108 RS-FEC, Clause 74 BASE-R FEC	• Used for Backplane connectivity. • Auto-negotiated options for use of no FEC, BASE-R FEC, or RS(528,514).
	25GBASE-CR	PMD: • IEEE 802.3 Clause 110 PCS • IEEE 802.3 Clause 107 PCS, Clause 108 RS-FEC, Clause 74 BASE-R FEC	• Used for connecting to DA Copper cable. • Auto-negotiated options for use of no FEC, BASE-R FEC, or RS(528,514).
	25GBASE-CR1/KR1	25G/50G Ethernet Consortium Schedule 3 v1.6.	• 25G CR/KR PMD as specified in 25G/50G Ethernet Consortium Schedule 3 v1.6. Compatible with IEEE 25G CR/KR except for auto-negotiation advertisement.

Table 3-20. Supported Electrical Modes [continued]

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 and active cables. • Appendix E linear mode for passive copper twin-ax.
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.
100M	SGMII	PMD: • SGMII spec PCS: • IEEE 802.3 Clause 36, 37 like	• Used for chip-to-chip connectivity.

3.2.2.2 MAC/PHY Interface

The E810 provides up to eight Ethernet ports, where there is a 1:1 mapping between the E810 MAC ONPI port and the PHY physical port. The interface modes are illustrated in the following diagrams.

3.2.2.2.1 MAC/PHY Interface Mapping

Figure 3-6 illustrates the default MAC-to-PHY connectivity. The controller core contains eight Ethernet MACs. The MACs are grouped into two sets of four. Within each set of four, one MAC supports all rates up to 100G (MAC0 and MAC1), a second MAC supports all rates up to 25G (MACs2 and MAC3), and the remaining MACs (MAC4-MAC7) support up to 10G.

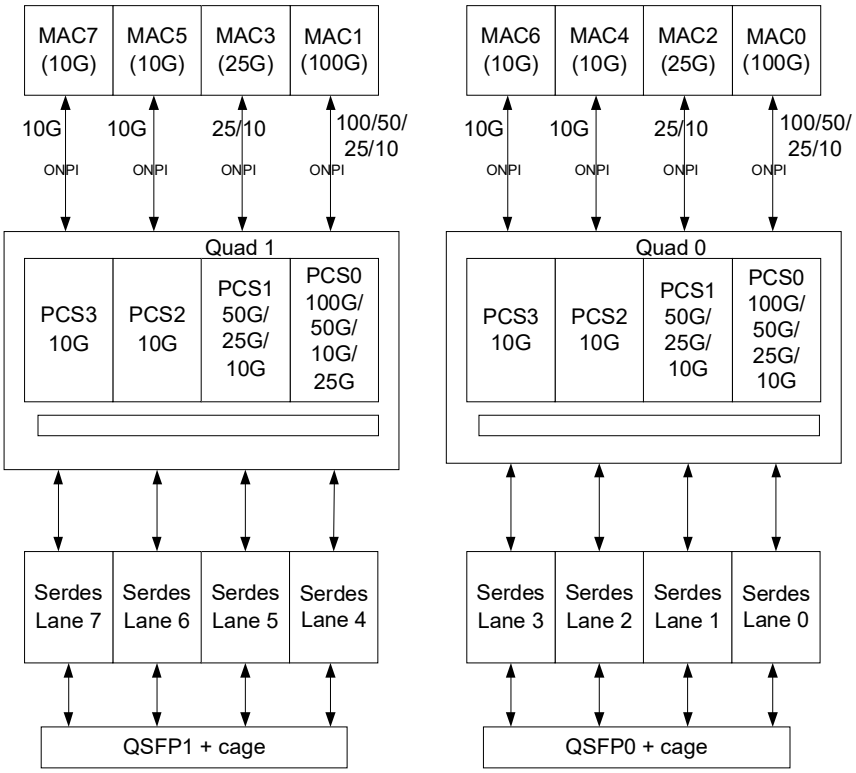


Figure 3-6. E810 - MAC-PCS-SerDes Default Mapping

Each set of four MACs is connected to a single PCS block that supports up to four ports. Within each PCS block is one PCS function (PCS0) that support all rates up to 100G, a second PCS function (PCS1) that supports rates up to 50G, and two PCS functions (PCS2/3) that support up to 10G.

When operating in dual port mode, MAC0 and MAC1 are connected to PCS0 of their respective PCS blocks, and the PCS distributes the data to up to four SerDes lanes needed for 4-lane modes, such as 100GBASE-CR4. This results in MAC0 utilizing SerDes lanes 0-3, while MAC1 utilizes SerDes lanes 4-7.

When operating in quad-port mode, only single lane per port modes are supported. MAC0 and MAC2 connect to PCS0 and PSC1 of the first PCS, while MAC1 and MAC3 connect to PCS0 and PCS1 of the second PCS block. Using the default PCS-to-SerDes mapping, this presents the data from MACs (0,2,1,3) on SerDes lanes 0, 1, 4, and 5 respectively.

To support a “breakout” mode where all four ports utilize the same SerDes quad, multiplexers in the PCS-to-SerDes path direct the data from both PCS blocks to a single SerDes quad. This mode allows a system design to support quad-port 25G in the same QSFP connector as a 100G port.

When working with lane rates of 50G, a 100G port is mapped to two lanes and 50G port is mapped to a single lane. As those rates are only supported by MAC0 and MAC1, both direct and breakout mappings are supported. While working at 100G port rates, the direct mapping maps MAC0 into lane0 and lane1, while MAC1 is mapped into lane4 and lane5. In 50G port rates, the direct mapping maps MAC0 to lane0 and MAC1 into lane4. In 50G port rates, additional breakout mapping can be applied, which maps MAC0 into lane0 and MAC1 into lane1.

The 50G “breakout” mode allows a system design to support two 50G ports using the same QSFP connector that was used when the port is configured to 100G.

3.2.2.3 Port and PMD Mapping

To manage the Ethernet links, E810 firmware must map the port number and PMD (SerDes lane) number along the way of the packet.

3.2.2.3.1 Port and PMD Nomenclature

The term "Logical port" to refers to an Ethernet port link as seen from the host perspective. A logical port is mapped to the E810 Ethernet MAC and to a PCIe physical function (PF). In addition, a logical port is mapped to a single or multiple PMD lanes of the internal PHY, depending on the PMD type. For link management admin commands or link topology commands, the E810 field refers to logical port.

E810 MAC port numbering might be different from logical port numbering, because of physical function mapping and the internal ONPI connection between the E810 controller core and the PHY.

3.2.2.3.2 Port Mapping

Table 3-21. Port Mapping

E810				PHY		
Physical Function (PF) Mapping			MAC Number	ONPI Port Number	PMD Lane Number[1]	
2-Port Default	4-Port Default	8-Port Default			Single Lane	Multi Lane
0	0	0	0	0	0	**0**,1,2,3
	1	1	2	2	1	1
		2	4	4	2	2
		3	6	6	3	3
1	2	4	1	1	4	**4**,5,6,7
	3	5	3	3	5	5
		6	5	5	6	6
		7	7	7	7	7

1. Bold identifies the auto-neg PMD lane

Note: E810 Ports 0-3 support up to 25G. There is a need to map those ports using the same lanes that were used for 100G, for breakout configurations. This mapping is described in Table 3-22.

Table 3-22. Port Mapping (4x25G on the First Quad)

Physical Function (PF)	E810 MAC Number	PHY ONPI Port Number	PMD Number
0	0	0	0
1	2	2	1
2	1	1	2
3	3	3	3

For 50G lane rates, E810 Ports 0-1 are mapped to lanes 0,1,4,5 using the following mapping:

Table 3-23. Port Mapping (50G Lane Rates)

Mode	Physical Function (PF)	E810 MAC Number	PHY ONPI Port Number	PMD Number[1]
2x100G	0	0	0	**0**,1
	1	1	1	**4**,5
2x50G	0	0	0	0
	1	1	1	4
2x50G Breakout	0	0	0	0
	1	1	1	2

1. Bold identifies the auto-neg PMD lane

3.2.2.3.3 Port Mapping - E810-XXVAM2 SKU

The E810-XXVAM2 product in the 21x21 mm package shares the same core features as the E810-CAM2/CAM1 products in the 25x25 mm package, with the exception of limited interfaces. The PCIe interface is limited to Gen4 x8, and the Ethernet interface limited to two ports maximum at 25G and lower speeds, or a single port of 50G, implemented as 2x25G SerDes (that is, 50GBASE-CR2).

Table 3-24. Port Mapping - E810-XXVAM2

Mode	Physical Function (PF)	E810 MAC Number	PHY ONPI Port Number	PMD Number[1]
2x25G	0	0	0	**0**
	1	1	1	**1**
1x50G	0	0	0	**0**,1

1. Bold identifies the auto-neg PMD lane

3.2.2.4 PHY Lane Mapping per Port Mode

In single-lane mode, there is a direct mapping between the PHY ONPI port and the PHY PMD lane. In multi-lane mode, a single PHY ONPI port is mapped to 2 or 4 PMD lanes. The E810 PHY Core supports multi-lane mode at port 0 and port 4. Port 0 can be mapped to PMD lanes 0-2 for a dual-lane mode, or lanes 0-3 for a 4-lane mode. Port 4 can be mapped to PMD lanes 2-3 for a dual-lane mode, or lanes 4-7 for a 4-lane mode.

Table 3-25. E810 PMD Lane Mapping per Port Mode

	Site #/PMD#/Supported PMD Rates									
Site	**Site 0**				**Site 1**				**Port Mode**	**Notes/Limitations**
PMD	**0***	**1**	**2**	**3**	**4***	**5**	**6**	**7**		
QSFP Lane	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**		
CFG1.1	50G		50G						2	**CFG1.1:** 4xSFP (2-port and 4-port modes) **Note:** The 4x SFP implementation also supports 2x50GBASE-R on PMD lane 0 and 2. This configuration multiplexes the Site 1 serial lanes to Site 0.
	25G	25G	25G	25G					4	
	10G	10G	10G	10G					4	
	1G	1G	1G	1G					4	
	100M	100M	100M	100M					4	
CFG3.1	100G-R4								2	**CFG3.1:** 1xQSFP (2-port and 4-port modes, including breakout) **Note:** This configuration multiplexes the Site 1 serial lanes to Site 0. Site 0 configuration now effectively supports QSFP and QSFP breakout cables. Site 1 is disabled due to the internal multiplexing structure.
	100G-R2								2	
	50G-R2		50G-R2						2	
	50G		50G						2	
	25G	25G	25G	25G					4	
	10G	10G	10G	10G					4	
	1G	1G	1G	1G					4	
	100M	100M	100M	100M					4	
CFG4.1	100G-R4				100G-R4				2	**CFG4.1:** 2xQSFP (2-port, 4-port, and 8-port modes) **Note:** CFG4.x is a superset of CFG3.x. A system designed with CFG4 can also use the breakout mode described in CFG3 on Site 0. **Note:** For the noted PMD lanes (Note 1), the configuration is only supported with topology resolution when breakout cables are plugged in. This would force the E810 to be configured in a 4-port/8-port mode.
	100G-R2				100G-R2				2	
	50G-R2				50G-R2				2	
	50G				50G				2	
	25G	25G[1]			25G	25G[1]			4	
	10G	10G[1]	10G[1]	10G[1]	10G	10G[1]	10G[1]	10G[1]	8	
	1G	1G[1]	1G[1]	1G[1]	1G	1G[1]	1G[1]	1G[1]	8	
	100M	100M[1]	100M[1]	100M[1]	100M	100M[1]	100M[1]	100M[1]	8	
CFG5	10G	10G	10G	10G					4	**CFG5:** 4x10GBASE-T (4-port mode)
	1G	1G	1G	1G					4	
	100M	100M	100M	100M					4	

Key:	100G (BASE-R4/-R2)	50G (BASE-R2/-R)	25G	10G	1G/100M

3.2.3 Link Management

3.2.3.1 Link Management Interfaces

The E810 supports either MDIO or I^2C interfaces for control plane connection between the MAC (master side) and external PHY devices. The MDIO or I^2C interfaces enable both MAC and firmware access to the PHY for monitoring and controlling of the PHYs functionality. E810 MDIO is compliant with the IEEE Std 802.3 Clause 45 as well as IEEE Std 802.3 Clause 22 frame formats and register address space for accessing legacy PHY devices. E810 I^2C is compliant with the I^2C bus specification.

Since I^2C and MDIO interfaces requires 3.3 V pads, the I^2C and MDIO controllers are not implemented within the E810. Instead they are placed in a widget. E810 firmware accesses the I^2C and MDIO controllers via registers.

The E810 supports up to six management interfaces (one per port, in 2-port or 4-port configurations) to control external PHY devices. In configurations where the number of ports is higher than four, each management interface should be shared (that is, a single management interface for a mezzanine card of four ports). When the protocol used is I^2C, an I^2C port replicator might be implemented on the 4-port mezzanine card. Depending on the PHY type to manage, the management interface can be configured for either MDIO or 2-wire management interface.

To manage multi-port PHYs, an I^2C/MDIO interface can be configured to control a quad-port PHY, or two I^2C/MDIO interfaces can be configured to control two dual-port PHYs. The PHY configuration registers for each port are mapped into the respective I^2C/MDIO address space. In configurations in which the number of ports is higher than four, five E810 I^2C/MDIO interfaces can be configured to control up to five quad-port PHYs or mezzanine cards with I^2C port replicator. The sixth I^2C/MDIO interface is usually shared between all devices and mezzanine cards and is used for PHYs, LED controllers, thermal sensor or other devices that can share the I^2C or MDIO bus. For example, in mezzanine card based topology, one I^2C port might be dedicated for each mezzanine card and one MDIO interface can be shared between all cards. The MDIO can be used for PHYs or other devices that can share the bus. The I^2C bus is either replicated, using I^2C port extender at the mezzanine card, or used for devices that can share the I^2C bus. Alternatively, the I^2C bus can be used by single I^2C device.

Note: PHY mezzanine daughter cards are not supported in the E810.

The E810 provides hardware acceleration of MDIO accesses over the 2-wire management interface. The device driver manages the external devices using the admin commands (see Section 3.2.4, "Link Configuration Admin Commands"). The device driver does not have direct access to the MDIO bus except for diagnostic purposes. Firmware performs direct access to MDIO interface, using register read and write commands to the widget, for reading and writing to the PHY device as described in the paragraphs that follow.

Figure 3-7 shows the a basic example of connectivity between the MAC and an external PHY/module for up to four ports.

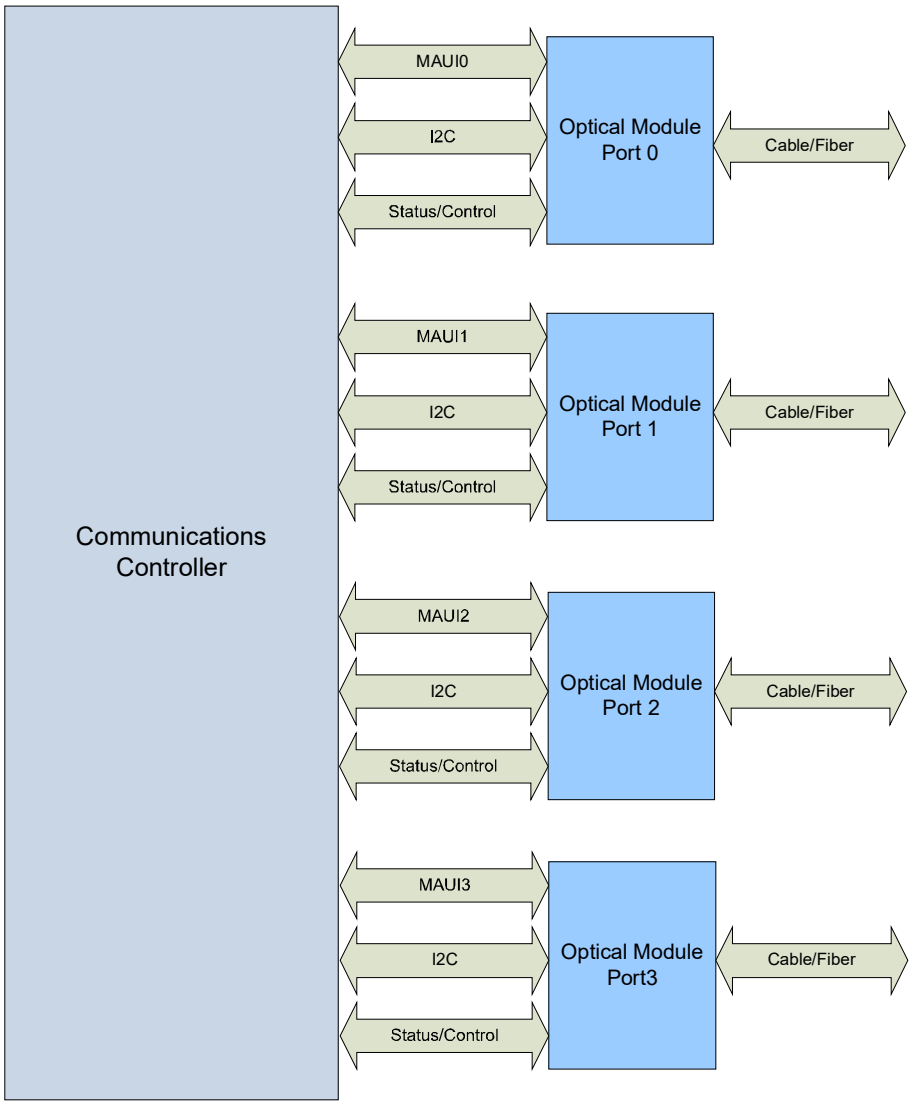


Figure 3-7. Basic PHY/Module Connectivity

3.2.3.2 Link Management Topologies

The E810 supports MDIO and 2-wire management interface (I^2C) for connectivity to external modules, re-timers, and PHYs (for example, SFP+ or QSFP+ optical and direct attached copper PHYs).

The E810 supports up to five management interfaces (one per port - for up to 4-port configurations) to control external PHY devices. Depending on the PHY type to manage these can be configured for either MDIO or 2-wire management interface.

The link topology is described using a netlist in NVM section (see Section 3.3).

Figure 3-8 and Figure 3-9 show the basic examples of connectivity between the MAC and an external module.

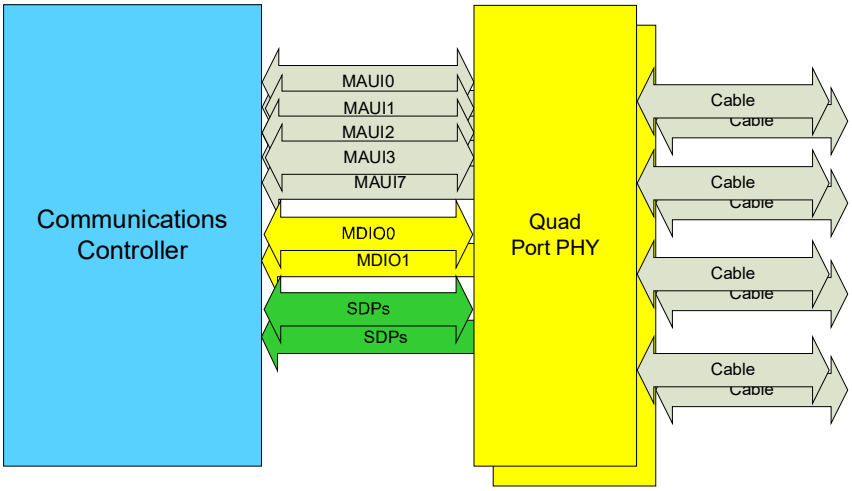


Figure 3-8. Basic Example of Module Connectivity

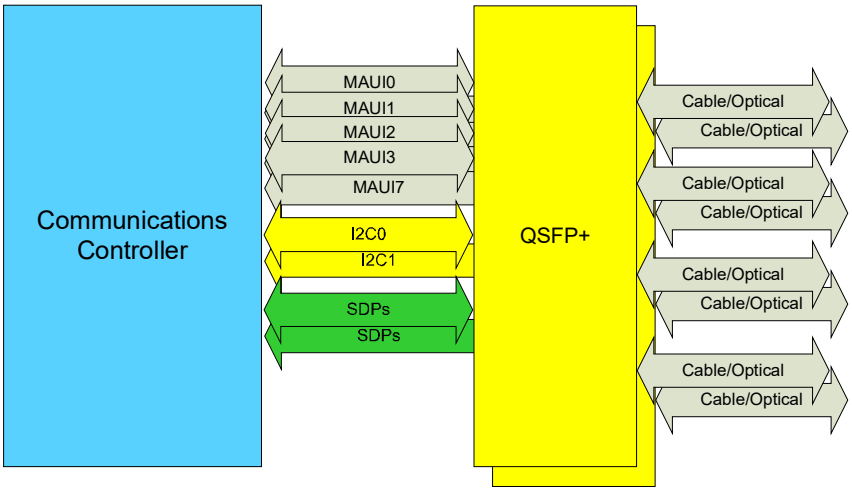


Figure 3-9. Basic Example of Module Connectivity

3.2.3.2.1 The E810's SFP+ Connectivity Scheme (Up to 4 Ports Topologies)

Figure 3-10 and Figure 3-11 illustrate the low speed digital signals used when connecting the E810 to SFP+ modules with topologies of up to four ports.

Note: The diagram illustrates the minimal connectivity required. Additional SDPs can be used to connect to the SFP+ module for direct hardware signaling (for example, *TX_FAULT*). However, these are not mandatory and can be replaced with register access through the I^2C management interface.

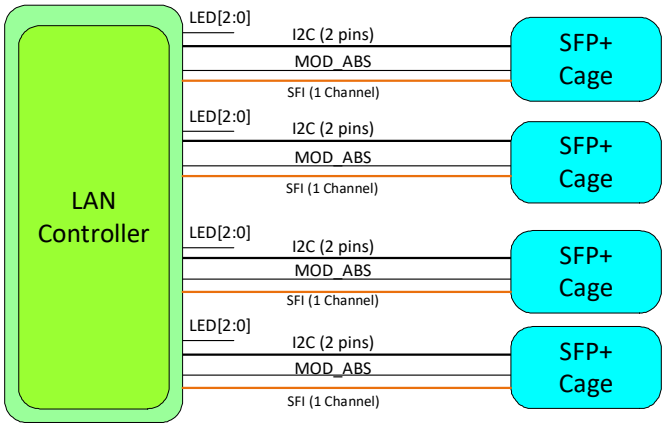


Figure 3-10. Quad 10G SFP+[1]

1. In 4-port configuration, the E810 provides up to 3 LEDs per port.

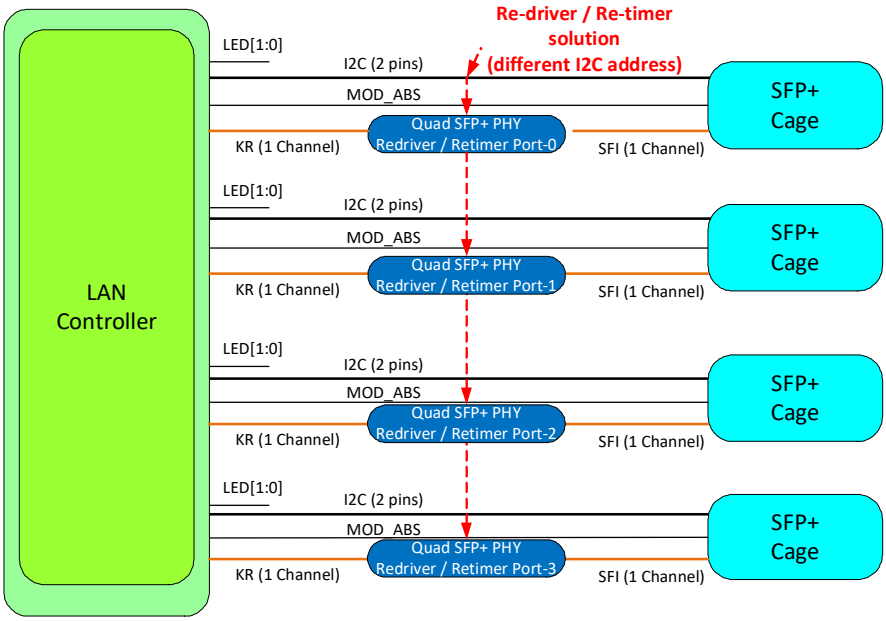


Figure 3-11. Quad 10G SFP+ with Re-driver PHY[1]

1. In 4 port configuration, the E810 provides up to 3 LEDs per port

3.2.3.2.2 The E810's QSFP+ Connectivity Scheme (Up to 8 Ports Topologies)

Figure 3-12 and Figure 3-13 illustrate the low speed digital signals used when connecting the E810 to QSFP+ modules.

Note: The diagrams illustrates the minimal connectivity required. Additional SDPs can be used to connect to the QSFP+ module for direct hardware signaling (for example, ResetL). However, these are not mandatory and can be replaced with register access through the I^2C management interface.

The E810 also supports a mode where an SFP module is inserted in a QSP+ cage using a QSA module adapter.

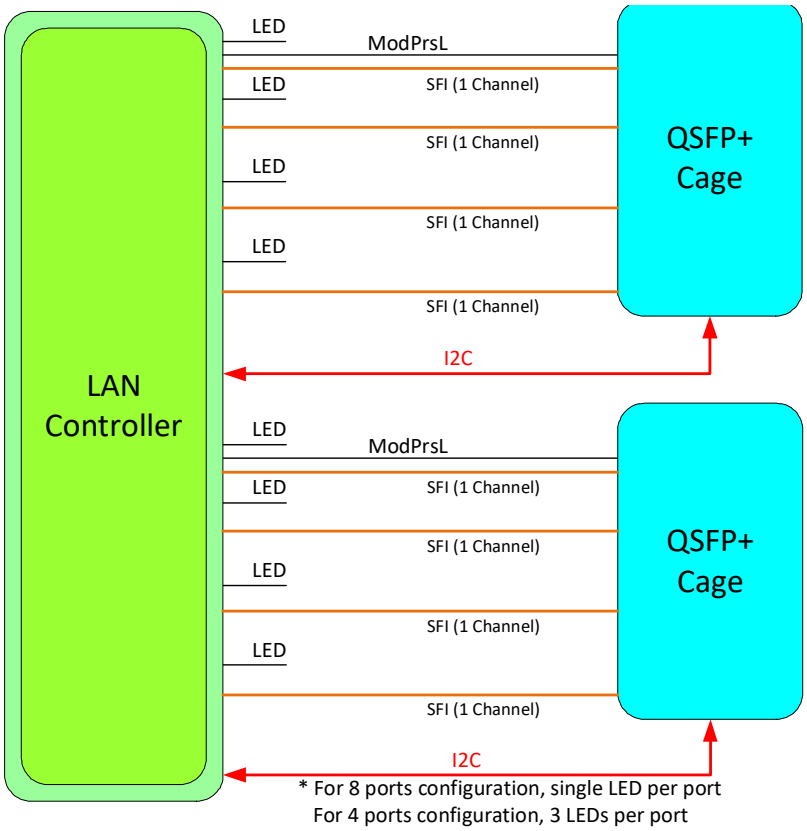


Figure 3-12. Octal 10GDual QSFP+

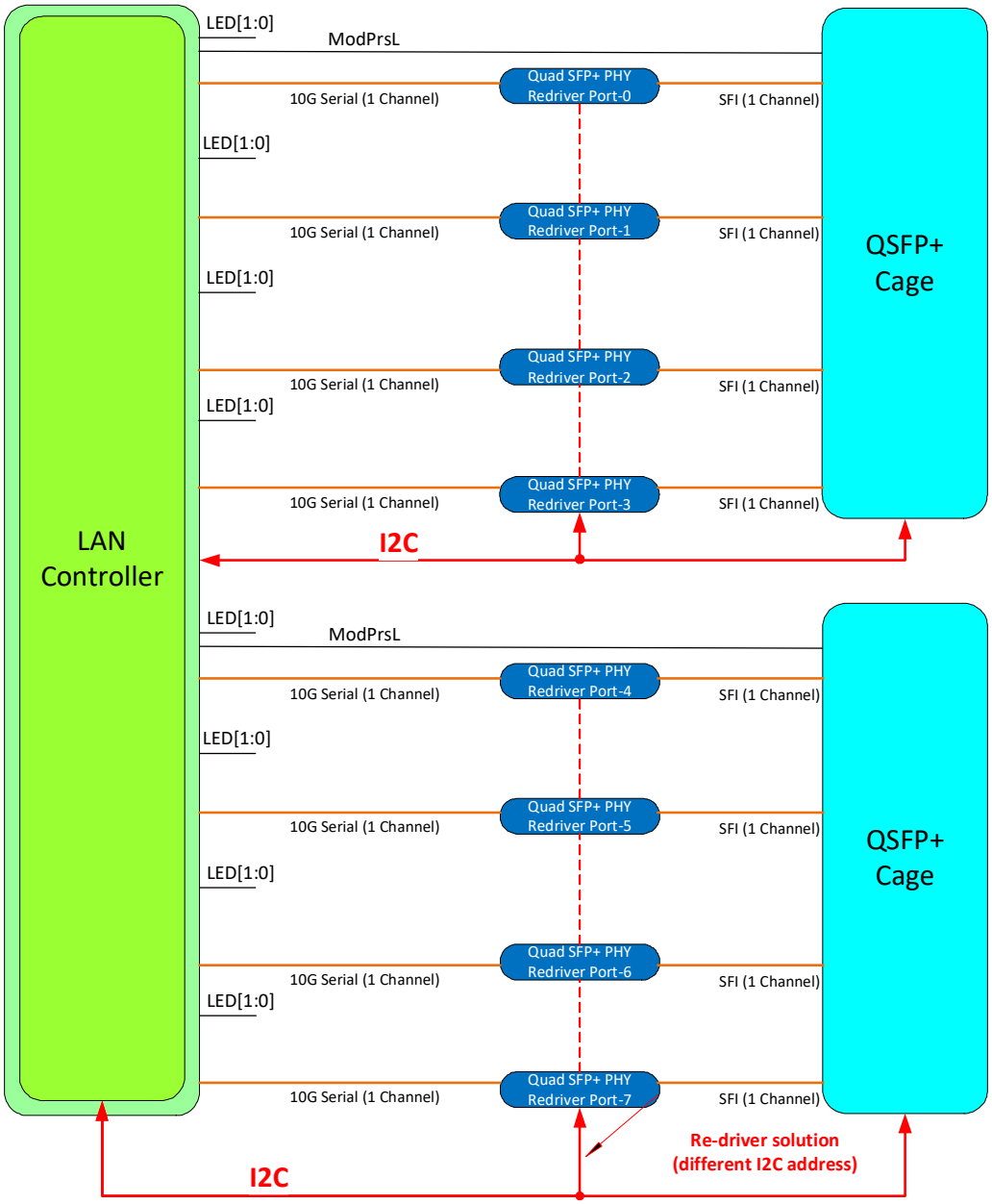


Figure 3-13. Octal 10GDual QSFP+ with Re-driver PHY

3.2.3.2.3 The E810's BASE-T Connectivity Scheme

Figure 3-14 illustrates the low speed digital signals used when connecting the E810 to BASE-T PHYs.

Note: The diagram illustrates the minimal connectivity required. Additional SDPs can be used to connect to the PHY for direct hardware signaling INT. However, these are not mandatory and can be replaced with register access through the MDIO management interface.

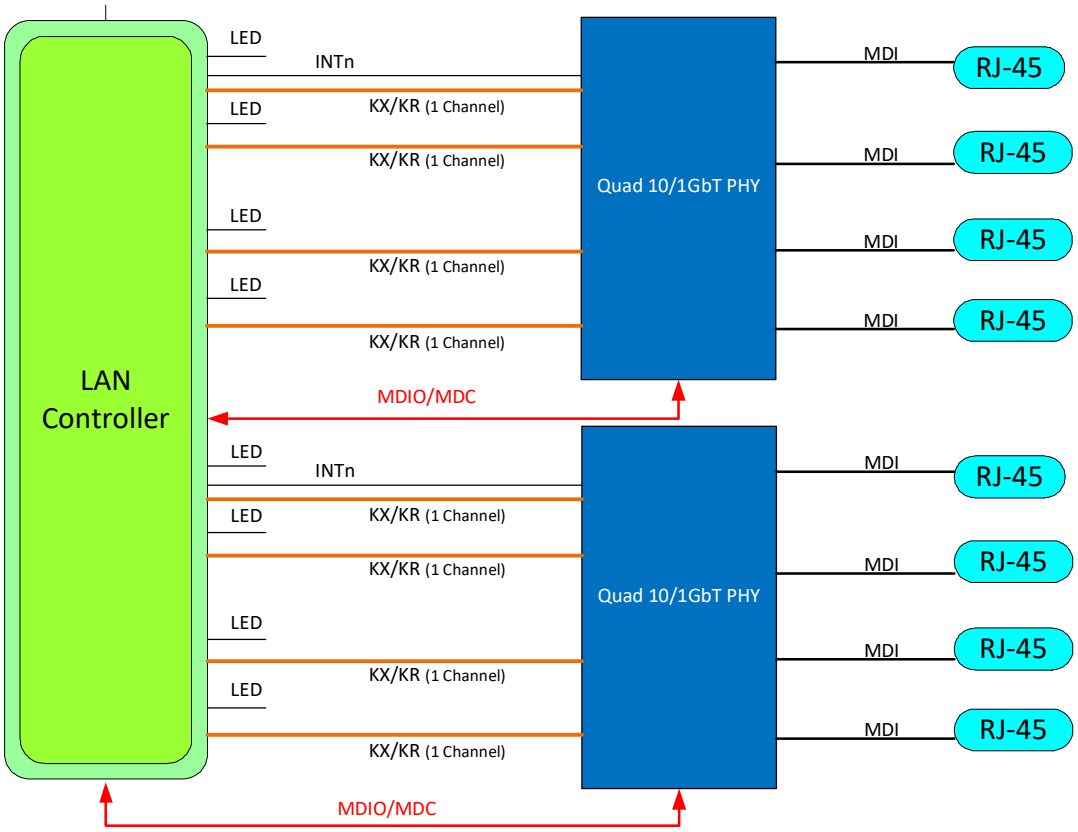


Figure 3-14. Octal 10G/1G BASE-T

Note: The SFP+ module is shown with four SDP pins per port for dedicated hardware functions. The *TX_Disable* bit of the SFP+ module is accessed by software through the I^2C interface. Alternatively, one of the global GPIO pins can be connected to *TX_Disable* pin.

Note: The SFP+ module is shown above with four SDPs. However, the minimum number of SDPs required is only one, which should be connected to the module's *Mod_ABS* pin.

Furthermore, when connecting to a pluggable 1GBASE-T module, the minimum number of SDPs required is two, since an additional SDP is required for connecting to the module *Rx_LOS* signal for accurate link state reporting to the link management firmware. There are, however, specific pluggable BASE-T modules that can be supported without *Rx_LOS*. For specific model numbers please contact Intel support.

3.2.4 Link Configuration Admin Commands

The E810 supports the following admin commands for configuring and managing the link. Software should use the admin commands to configure the link. This includes configuring the MAC and internal/external PHY devices. The firmware provides link configuration and status services to the device driver based on these admin commands.

Software can use the Get Link Status command to find out the current status of the link.

Table 3-26. Link Configuration Admin Commands (0x06xx)

Command	Opcode	Description	Section Reference
Set PHY Config	0x0601	Sets various PHY configuration parameters on the port.	3.2.4.1.1
Set MAC Config	0x0603	Sets various MAC configuration parameters on the port.	3.2.4.1.2
Setup Link and Restart Auto-Negotiation[1]	0x0605	Sets up the link and restarts link establishment. This command applies the latest link configuration as configured with the Set PHY Config (0x0601) command. The command also restarts the link establishment process. For links configured for auto-negotiation, it starts the auto-negotiation process. This operation could bring down the link. This command must be executed to allow the latest link parameters to take effect on the link.	3.2.4.1.3
Get PHY Abilities	0x0600	Gets various PHY abilities supported on the port.	3.2.4.1.4
Get Link Status	0x0607	Gets link status of the port.	3.2.4.1.5
Link Status Event	0x0607	Firmware sends this asynchronous event notification to software when there is a change in status in any of the event causing conditions (such as link up/down or other link error conditions).	3.2.4.1.6
Set Event Mask	0x0613	Sets event mask. Software can mask some or all of the link status event causing conditions.	3.2.4.1.7

1. In SFP mode, the Setup Link and Restart Auto-Negotiation command must be executed by the device driver or any other change in link parameters to take effect on the link. This operation could disrupt the link since the link state might toggle while the link is re-initialized with the new parameters.

3.2.4.1 Link Configuration Commands

This section provides a detailed description of the link configuration admin commands and their structure.

3.2.4.1.1 Set PHY Config (0x0601)

This command is used by the device driver to set the various PHY configuration parameters supported on the port.

This is a indirect command. Set PHY Config command parameters data structure is placed in the data buffer.

Note: This command must be followed by the Setup Link and Restart Auto-Negotiation (0x0605) command for any changes to the link parameters to actually take place. Some parameter settings might take effect immediately with this command execution, as specified below.

Table 3-27. Set PHY Config Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0601	Command opcode.
Datalen	4-5	0x18	Length of the data buffer in bytes.
Return Value/VFID	6-7		Return value. Zeroed by device driver. Written by firmware. 0 = No error. Command completed successfully. 21 = EMODE. 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. At this case, the link is not configured.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Logical Port Number	16		Logical Port number. This field specifies the logical port number, and is ignored in the E810
Reserved	17-23	Reserved	Value 0x0.
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.

In the E810, the command is indirect and contains a data buffer. The structure of the data buffer is described in Table 3-28.

Table 3-28. Set PHY Config Command Data Structure

Name	Bytes.Bits	Value	Remarks
PHY Type	0-15	PHY Type	128 bits PHY type (PHY capabilities) supported on port. 128 bits structure, describing capabilities of the outermost PHY, with one bit per PHY capability. PHYs connected to the E810 might be capable of supporting multiple PHY types, indicated using the link topology netlist The software driver uses this command to indicate the capabilities to enable. This structure is identical to the extended PHY capabilities structure, as described in Section 3.3.3.2.1. This parameter is used by the device driver to set the various PHY capabilities 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 Abilities (0x0600) command. When the link establishment state machine (LESM) is enabled, the E810 tries one of the enabled PHY types that matches the media capabilities and selects this PHY capability when the link is up. When the LESM is disabled, this field should enable only a single value, and then the E810 is forced to operate in that selected mode. Some of the PHY capabilities might require the auto-negotiation feature, which might resolve some of the link parameters, such as speed and FEC, according to the negotiation result with the link partner.
PAUSE Ability	16.0-16.1	Pause Ability	**Bit 16.0:** Tx Link Pause 0b = Disable PAUSE ability. 1b = Enable IEEE 802.3x Tx link PAUSE ability. **Bit 16.1:** Rx Link Pause 0b = Disable PAUSE ability. 1b = Enable IEEE 802.3x Rx link PAUSE ability. Auto-negotiation might have be restarted for this configuration to take effect over the link. This parameter is used by the device driver to set the IEEE 802.3x pause ability of the port. The E810's pause ability can be read using the Get PHY Abilities (0x0600) or Get Link Status (0x0607) commands. The device driver might disable the IEEE 802.3x link PAUSE ability using this command. If the link is already up and configured, the device driver needs to restart auto-negotiation, so the updated PAUSE ability could be advertised to the link partner for the setting to take effect on the link. ***Note:*** When PFC is enabled, firmware turns off the PAUSE ability bits during auto-negotiation regardless of the setting used in the admin command. However, if PFC will be disabled in the future, the last setting of the PAUSE ability bits is used.
Low Power Ability	16.2	Low Power Mode	**Bit 16.2:** Power Mode 0b = High Power mode. 1b = Low Power mode. QSFP+: • This is ignored if NVM loaded <Low Power Ability> = Low Power. When QSFP+ is shared between multiple ports: • Firmware sets the power mode based on the setting of the first (lower power number) port in the group and ignore the settings of the other ports in the group. • The setting of Low Power Ability by one PF automatically changes the Low Power Ability of all PFs sharing the QSFP+. This can result in link loss on all ports. • Firmware identifies which ports are sharing a QSFP+ by looking at the ModPresL SDP and seeing that it is shared between multiple ports.
Enable Link	16.3	Enable Link	**Bit 16.3:** Link Enable 0b = Disable the link. 1b = Enable the link. Device driver should not force link down when port is being used for manageability or WoL.

Table 3-28. Set PHY Config Command Data Structure [continued]

Name	Bytes.Bits	Value	Remarks
Reserved	16.4	Reserved	Reserved.
Enable Automatic Link Update	16.5	Enable Automatic Link Update	**Bit 16.5:** Automatic Link Update Enable 0b = The device driver maintains the responsibility for sending the Setup Link and Restart Auto-Negotiation (0x0605) command. 1b = Firmware automatically executes the Setup Link and Restart Auto-Negotiation (0x0605) command following this command. When automatic link update is enabled, the device driver should be aware that a link change event can occur following the Set PHY Config (0x0601) command.
LESM Enable	16.6	LESM Enable	**Bit 16.6:** LESM Enable 1b = Enables the Link Establishment State Machine.
Auto FEC Enable	16.7	Auto FEC Enable	**Bit 16.7:** Auto FEC Enable 1b = Enables the automatic selection of FEC mode.
Low Power Control	17	D3cold LPAN	**Bit 17.0:** D3cold LPAN 0b = Disable D3cold low power auto-negotiation. 1b = Enable D3cold low power auto-negotiation. **Bit 17.3:** AN37 Enable - Clause 37 auto-negotiation enable. 0b = AN37 disabled. 1b = AN37 enabled on 1G links. ***Note:*** The value of *AN37 Enable* bit should be propagated from Get PHY Abilities (0x0600) output to Set PHY Config (0x0601) input for all default Software and firmware flows. **All other bits** = Reserved. Must be zero.
EEE Capability Enable	18-19	EEE Capability	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 the E810 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. See Section 3.3.6.6.8. **Bit 18.0:** EEE is enabled for 100BASE-TX. **Bit 18.1:** EEE is enabled for 1000BASE-T. **Bit 18.2:** EEE is enabled for 10GBASE-T. **Bit 18.3:** EEE is enabled for 1000BASE-KX. **Bit 18.4:** EEE is enabled for 10GBASE-KR. **Bit 18.5:** EEE is enabled for 25GBASE-KR **Bit 18.7:** EEE is enabled for 50GBASE-KR2 **Bit 19.0:** EEE is enabled for 50GBASE-KR-PAM4 **Bit 19.1:** EEE is enabled for 100GBASE-KR4 **Bit 19.2:** EEE is enabled for 100GBASE-KR2-PAM4 **All other bits** = Reserved. 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 E810 can be read by the Get PHY Abilities (0x0600) command. The device driver might set EEE capability for a subset of PHY types supported by the E810.
EEER	20-21	EEER Value	Value to program the EEER register. ***Note:*** The EEER register is changed immediately with the execution of this command.

Table 3-28. Set PHY Config Command Data Structure [continued]

Name	Bytes.Bits	Value	Remarks
Link FEC Options	22	Link FEC Options	Sets the FEC options to the link. This field is aligned with the FEC options in "Link options 0" of the topology netlist. **Bit 22.0:** FIRE_CODE_10_ABILITY Enable Controls FEC capability advertisement for 10G KR. 0b = FEC disabled. 1b = FEC enabled (advertised). **Bit 22.1:** FIRE_CODE_10_REQUEST Controls FEC capability request for 10G KR. 0b = FEC disabled. 1b = FEC Enabled (requested). **Bit 22.2:** RS_528_REQUEST Controls RS FEC 528 capability request for 25G lanes KR/KR-S/KR1/CR/CR-S/CR1. 0b = FEC disabled. 1b = FEC Enabled (requested). **Bit 22.3:** FIRE_CODE_25_REQUEST Controls KR FEC capability request for 25G lanes KR/KR-S/KR1/CR/CR-S/CR1. 0b = FEC disabled. 1b = FEC Enabled (requested). **Bit 22.4:** RS_544_REQUEST Controls RS FEC 544 capability request for 25G/50G lanes KR/KR-S/KR1/CR/CR-S/CR1. **Bit 22.5:** Reserved **Bit 22.6:** RS_528_ABILITY Controls RS FEC ability advertisement for 25G KR1/CR1. 0b = FEC disabled. 1b = FEC enabled (advertised). **Bit 22.7:** FIRE_CODE_25_ABILITY Controls KR FEC ability advertisement for 25G KR1/CR1. 0b = FEC disabled. 1b = FEC enabled (advertised).
Module Compliance Enforcement	23.0		0b = Lenient 1b = Strict
Reserved	23.1-23.7	0x0	Reserved. Set to 0x0.

The following structure describes the response by firmware to the Set PHY Config command.

Table 3-29. Set PHY Config Command Response

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0601	Command opcode.
Datalen	4-5		Length of the data buffer in bytes.
Return Value/VFID	6-7		Return value. 0x0 = Command success. Returns EPERM code if the operation is not permitted.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Logical Port Number	16		Logical Port number. This field specifies the logical port number, and is ignored in the E810.
Param0	17-19	Reserved	Zeroed by firmware,. Value is ignored.
Param1	20-23	Reserved	Must be 0x0. Value is ignored.
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.

Note: When *Enable Automatic Link Update* is set to 1b, firmware sends a completion for the Set PHY Config command only after making all the necessary configuration changes and executing the Setup Link and Restart Auto-Negotiation (0x0605) command.

3.2.4.1.2 Set MAC Config (0x0603)

This command is used by the device driver to set the various MAC configuration parameters supported on the port. This status is indicated by the command response.

This is a direct command. The Set MAC Config command parameters data structure is placed in the command descriptor.

Table 3-30. Set MAC Config Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0603	Command opcode.
Datalen	4-5	0x0	Must be 0x0. Value is ignored.
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 firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Set MAC Config	16-31	See Table 3-31	16-byte data structure that holds the Set MAC Config command parameters as listed in Table 3-31.

Table 3-31 lists the data structure of the Set MAC Config command parameters, such as max frame size, and so on.

Table 3-31. Set MAC Config Command Data Structure

Name	Bytes.Bits	Value	Remarks
Max Frame Size	0-1	Max Frame Size	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	2.0-2.2	Reserved	Must be 0x0
Pacing config	2.3-2.7	Pacing Config	**Bits 2.6-2.3:** This is 4 bit field that allows configuring PACE parameter in the MAC to slow down the effective data rate. For data rate pacing, as defined in Table 3-18. For Fixed IPG pacing, defines the number of 16-byte words added. **Bit 2.7:** Pacing Type 0b = Data rate pacing. 1b = Fixed IPG pacing.
Transmit Timer Priority	3	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. For additional register description, see Section 3.2.1.5.

Table 3-31. Set MAC Config Command Data Structure [continued]

Name	Bytes.Bits	Value	Remarks
Transmit Timer Value	4-5	Transmit Timer Value	This is the priority <n> timer value that is included in the XOFF frames being transmitted, where <n> is selected in the *Transmit Timer Priority* bitmap above. <n> = 0 is used for Link Level FC. For additional register description, see Section 3.2.1.5.
FC Refresh Threshold	6-7	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 *Transmit Timer Priority* 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. For additional register description, see Section 3.2.1.5.
Auto Drop Blocking Packets	8.0		This bit 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	8.1-15	Reserved	Must be 0x0.

The following data structure describes the response by firmware to the Set MAC Config command.

Table 3-32. Set MAC Config Command Response

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0603	Command opcode.
Datalen	4-5	0x0	Must be 0x0. Value is ignored.
Return Value/VFID	6-7		Return value. 0x0 = Command success. Returns EPERM code if the operation is not permitted.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-31	Reserved	Value 0x0.

3.2.4.1.3 Setup Link and Restart Auto-Negotiation (0x0605)

This command is used by the device driver to setup the link and execute previously-sent Set PHY Config (0x0601) commands, as well as restart the link establishment process. When the link is configured to auto-negotiation mode, the link establishment state machine also starts the auto-negotiation process over the link. This command must be executed for any change in link parameters, such as set link speed (and so on) to take effect.

This is a direct command.

Table 3-33. Restart Auto-Negotiation Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0605	Command opcode.
Datalen	4-5	0x0	Must be 0x0. Value is ignored. 0 = No error. Command completed successfully. 20 = EMODE. 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. At this case, the link is not configured.
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 firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Logical Port Number	16		Logical Port number. This field specifies the logical port number, and is ignored in the E810.
Reserved	17	Reserved	Must be 0x0. Value is ignored.
Command Flags	18	Command	**Bit 18.1:** Restart Link 1b = Restart the link. **Bit 18.2**[1]**:** Enable Link 0b = Disable link. 1b = Enable link. **All other bits** = Reserved. Must be zero. This command might be executed automatically by firmware, following a Set PHY Config (0x0601) command, with the *Enable Automatic Link Update* bit (16.5) set. In such a case, these bits are assigned by firmware as follows: Bit 18.1 is set to 1. ***Note:*** Bit 18.2 is copied from the *Enable Link* field in the Set PHY Config command.
Reserved	19-31	Reserved	Must be 0x0. Value is ignored.

1. Used by the device driver to enable/disable the link without modifying the other link settings. This is useful at POR when an application needs to have link powered down until the device driver loads.

The following structure describes the response by firmware to the Restart Auto-Negotiation command.

Table 3-34. Restart Auto-Negotiation Command Response

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0605	Command opcode.
Datalen	4-5	0x0	Must be 0x0. Value is ignored.
Return Value/VFID	6-7		Return value. 0x0 = Command success.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Logical Port Number	16		Logical Port number. This field specifies the logical port number, and is ignored in the E810.
Reserved	17-31	Reserved	Value 0x0.

3.2.4.1.4 Get PHY Abilities (0x0600)

This command is used by the device driver to find out the various PHY abilities supported on the port.

This is an indirect command.

Table 3-35. Get PHY Abilities Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0600	Command opcode.
Datalen	4-5		Length of the response buffer in bytes. The software driver should supply a buffer with enough space to be filled by firmware
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 firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Logical Port Number	16		Logical Port number. This field specifies the logical port number, and is ignored in the E810.
Reserved	17	Reserved	Must be 0x0. Value is ignored.
Param0	18-19	Command	First command parameter. **Bit 18.0:** Report Qualified Modules List of qualified modules is part of the response only when this bit is set to 1b. **Bits 18.1-18.2:** Report Mode 00b = **Report Topology Capabilities, without Media:** Report capabilities structure as read from the topology netlist in NVM or ID EEPROM. This report ignores the media that might be connected and limits the capabilities. 01b = **Report Topology Capabilities, with Media:** Report capabilities as read from topology netlist in NVM and ID EEPROM, intersect with the media information, if exists Examples: When the topology includes cage, but it is empty, it reports all possible module types and speeds, just like 00b report mode. When there is specific media connected (that is, a module is populated into the cage), it reports only modes speeds supported by the media or the module. 10b = **Report Active Configuration:** Report the capabilities of the active configuration. The active configuration can be taken from the last software configuration, but it can be modified by firmware (for example when a new module is inserted or NCSI command is executed. When the Override Enable bit is set at the Link Default Override Mask PFA section, the capabilities are taken from the same PFA section. 11b = Reserved. **All other bits** = Reserved.
Reserved	20-23	Reserved	Must be 0x0. Value is ignored.
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.

Table 3-36 lists the Get PHY Abilities response structure returned by firmware to the Get PHY Abilities command. The response, opcode, and Get PHY Abilities response data structure buffer address are placed in the descriptor. The Get PHY Abilities response data structure is placed in a buffer.

Table 3-36. Get PHY Abilities Command Response

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0600	Command opcode.
Datalen	4-5		Set by firmware according to the actual size of the response buffer.
Return Value/VFID	6-7		Return value. EINVAL = Invalid parameters (for example, buffer too small). EIO = Error while accessing the information. EAGAIN = PHY/module interface currently busy, retry.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Logical Port Number	16		Logical Port number. This field specifies the logical port number, and is ignored in the E810.
Reserved	17-23	Reserved	Reserved. Must be set to 0x0.
Data Address High	24-27	Buffer Address	Buffer address. Buffer content is listed in Table 3-37.
Data Address Low	28-31	Buffer Address	

Table 3-37 lists the format of the buffer content for the Get PHY Abilities reply.

Table 3-37. Get PHY Abilities Command Response Data Structure

Name	Bytes.Bits	Value	Remarks
PHY Type	0-15	PHY Type	See detailed description in Table 3-46.
Pause Ability	16.0:16.1	Pause Ability	**Bit 16.0:** Returns 1b if the port advertises IEEE 802.3x Tx link PAUSE ability. Otherwise, returns 0b. **Bit 16.1:** Returns 1b if the port advertises IEEE 802.3x Rx link PAUSE ability. Otherwise, returns 0b. This parameter can be used by the device driver to determine the IEEE 802.3x PAUSE abilities of the port.
Low Power Ability	16.2	Low Power Ability	**Bit 16.2:** Power Mode 0b = High Power mode. 1b = Low Power mode.
Link Mode	16.3	Link Mode	**Bit 16.3:** Link Mode 0b = Link is disabled. 1b = Link is enabled.
Reserved	16.4		Reserved.
Enable Module Qualification	16.5	Enable Module Qualification	**Bit 16.5:** Module Qualification Returns 1b if a external module or PHY qualification check is enabled.
LESM Enable	16.6	LESM Enable	**Bit 16.6:** LESM Returns 1b if the Link Establishment State Machine is enabled.
Auto FEC Enable	16.7	Auto FEC Enable	**Bit 16.7:** Auto FEC Returns 1b if the automatic selection of FEC mode is enabled.

Table 3-37. Get PHY Abilities Command Response Data Structure [continued]

Name	Bytes.Bits	Value	Remarks
Low Power Control / AN Modes	17	D3cold LPAN	**Bit 17.0:** D3cold LPAN 0b = D3cold low power auto-negotiation disabled. 1b = D3cold low power auto-negotiation enabled. **Bit 17.1:** 0b = AN28 disabled. 1b = AN28 enabled. **Bit 17.2:** 0b = AN73 disabled. 1b = AN73 enabled. **Bit 17.3:** 0b = AN37 disabled. 1b = AN37 enabled on 1GF PHY Types. **All other bits =** Reserved. Must be zero. ***Note:*** The auto-negotiation enable bits are reserved when Get PHY Abilities (0x0600)" is called with the *Report Active Configuration* parameter indicated, and firmware should return these bits as 0.
EEE Capability	18-19	EEE Capability	Reports EEE capability for each PHY type supported on the port. One bit per PHY type. These capabilities refers to the outermost PHY connected to the E810 link. The following parameter indicates the bit number. This structure is aligned with the "EEE options 1", in the link topology netlist. See Section 3.3.6.6.8. **Bit 18.0:** EEE is enabled for 100BASE-TX. **Bit 18.1:** EEE is enabled for 1000BASE-T. **Bit 18.2:** EEE is enabled for 10GBASE-T. **Bit 18.3:** EEE is enabled for 1000BASE-KX. **Bit 18.4:** EEE is enabled for 10GBASE-KR. **Bit 18.5:** EEE is enabled for 25GBASE-KR **Bit 18.7:** EEE is enabled for 50GBASE-KR2 **Bit 19.0:** EEE is enabled for 50GBASE-KR-PAM4 **Bit 19.1:** EEE is enabled for 100GBASE-KR4 **Bit 19.2:** EEE is enabled for 100GBASE-KR2-PAM4 **All other bits** = Reserved, Must be zero.
EEER	20-21	EEER Value	Content of EEER register.
Current PHY ID/Vendor OUI	22-25	PHY ID/OUI	This parameter is used by the device driver to find out the PHY/module ID connected on the port. If the E810 is connected to an external BASE-T PHY: This 4-byte field returns the {OUI, Manufacturer Model#, Revision ID} as defined in IEEE 802.3, 22.2.4.3.1 PHY Identifier (Registers 2 and 3). **Bytes 23:22:** Register3. **Bytes 25:24:** Register2. If the E810 is connected to an external module: This field returns the three-byte vendor OUI of the module (MSB is padded with zeros).
Current PHY FW version	26-33	PHY Firmware Version	This 8-byte parameter indicates the external outermost PHY firmware version

Table 3-37. Get PHY Abilities Command Response Data Structure [continued]

Name	Bytes.Bits	Value	Remarks
Link FEC Options	34	Link FEC Options	Returns the FEC options of the link. This field is aligned with the FEC options in "Link options 0" of the topology netlist. See Section 3.3.6.6.5. **Bit 34.0:** FIRE_CODE_10_ABILITY Enable Controls FEC capability advertisement for 10G KR. 0b = FEC Disabled. 1b = FEC Enabled (advertised). **Bit 34.1:** FIRE_CODE_10_REQUEST Controls FEC capability request for 10G KR. 0b = FEC Disabled. 1b = FEC Enabled (requested). **Bit 34.2:** RS_528_REQUEST Controls RS FEC 528 capability request for 25G KR/KR-S/KR1/CR/CR-S/CR1. 0b = FEC Disabled. 1b = FEC Enabled (requested). **Bit 34.3:** FIRE_CODE_25_REQUEST Controls KR FEC capability request for 25G KR/KR-S/KR1/CR/CR-S/CR1. 0b = FEC Disabled. 1b = FEC Enabled (requested). **Bit 34.4:** RS_544_REQUEST Controls RS FEC 544 capability request for 25G KR/KR-S/KR1/CR/CR-S/CR1. **Bit 34.5:** Reserved **Bit 34.6:** RS_528_ABILITY Controls RS FEC ability advertisement for 25G KR1/CR1. 0b = FEC Disabled. 1b = FEC Enabled (advertised). **Bit 34.7:** FIRE_CODE_25_ABILITY Controls KR FEC ability advertisement for 25G KR1/CR1. 0b = FEC Disabled. 1b = FEC Enabled (advertised).
Module Compliance Enforcement	35.0		0b = Lenient 1b = Strict
Reserved	35.1-35.7	Reserved	Reserved.
Current Module Extended Compliance Code	36	Extended Compliance Code	Returns the extended compliance code of the module as defined in SFP+ (Address 0xA0, Byte 36) specification. This parameter is used by the device driver to find out the module type on the port when connected to external modules.

Table 3-37. Get PHY Abilities Command Response Data Structure [continued]

Name	Bytes.Bits	Value	Remarks
Current Module Type	37-39	Module Type	Returns the three-byte module ID. First byte: Module identifier. Defined by SFP+ (Address 0xA0, Byte 0) or QSFP+ (Address 128, page 0) specifications. Second byte: The following bits might be set to indicate the supported technologies: 0 = SFP+ Cu Passive 1 = SFP+ Cu Active 4 = 10G BASE-SR 5 = 10G BASE-LR Remaining bits are reserved. Third byte: GbE compliance code. Defined by SFP+ (Address 0xA0, Byte 6) or QSFP+ (Address ess134, page0) specifications. This parameter is used by the device driver to find out the module type on the port when connected to external modules. For example, the E810 might be connected to an SFP+ or QSFP+ optical or direct attached copper modules. The format of the module type returns the ID and Ethernet compliance code fields as defined in the SFP+ or QSFP+ specifications. There is no separate Ethernet compliance code for SFP+ copper modules. It is reported in a separate byte in SFP+ module. However, the E810 uses the unused bits in second byte to report SFP+ direct attach cables.
Qualified Module Count	40		Number of qualified modules to be listed in the following bytes.
Reserved	41-47	Reserved	Reserved.
Qualified Module ID-n	48+n*32 - 79+n*32		This is a list of qualified modules that are supported by the E810 and might be connected. The list contains a 24-byte field per module, based on IEEE Std 802.3 definition of device ID, containing: • Vendor OUI (3 bytes). • Reserved (1 bytes). • Vendor Part# (16 bytes). • Vendor Rev# (4 bytes). • Last 8bytes are reserved.

3.2.4.1.5 Get Link Status (0x0607)

This command is used by the device driver to find out the link status of the port. Firmware returns link status = 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.

In the E810 this is an indirect command, and returns the Get Link Status information structure within the buffer.

Table 3-38. Get Link Status Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0607	Command opcode.
Datalen	4-5		Length of the data buffer in bytes.
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 firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Logical Port Number	16		Logical Port number. This field specifies the logical port number, and is ignored in the E810.
Reserved	17	Reserved	Must be 0x0. Value is ignored.
Command Flags	18-19	Command Flags	**Bits 18.1-18.0:** 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. See Section 3.2.4.1.6 for details on LSE and enabling/disabling LSE events. **All other bits** = Reserved. Must be 0x0. Value is ignored.
Reserved	20-23	Reserved	Must be 0x0. Value is ignored.
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 structure describes the response by firmware to the Get Link Status command.

Table 3-39. Get Link Status Command Response

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0607	Command opcode.
Datalen	4-5		Length of the data buffer in bytes.
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 firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Logical Port Number	16		Logical Port number. This field specifies the logical port number, and is ignored in the E810.
Reserved	17	Reserved	Must be 0x0. Value is ignored.
Command Flags	18-19	Command Flags	**Bit 18.0:** LSE Enable Enable link status event notification to software. Firmware sets this bit to 1b to indicate that LSE is enabled, or sets to 0b if LSE is disabled. See Section 3.2.4.1.6 for further details on LSE and enabling/disabling LSE events. **All other bits** = Reserved. Must be 0, value is ignored.
Reserved	20-23	Reserved	Must be 0x0. Value is ignored.
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.

Table 3-40 lists the data structure of the Get Link Status command parameters, such as link up/down, negotiated/operating speed, fault conditions, and so on.

Table 3-40. Get Link Status Command Response Data Structure

Name	Bytes.Bits	Value	Remarks
Topology/Media Conflicts	0		Topology and media conflict reporting. **Bit 0.0:** 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. **Bit 0.1:** Unresolved media conflict detects. 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. **Bit 0.2:** LOM Topology netlist corrupted. The LOM topology netlist is corrupted and cannot be read from the NVM. **Bit 0.3:** Reserved. **Bit 0.4:** Topology netlist load detected Unreachable port. **Bit 0.5:** Topology netlist load detected Underutilized port. **Bit 0.6:** Topology netlist load detected Underutilized media. **Bit 0.7:** Unsupported Media detected.
Link Configuration Error	1		**Bit 1.0:** Link Configuration Error. Link Configuration Error was detected during attempt to bring the link up. **All other bits** = Reserved.
Link Status	2.0	Function Link Status	**Bit 2.0:** Function Link Status Returns 1b if link status = up, or returns 0b if the link status = down. This parameter indicates if the Link is up and ready for data communication.

Table 3-40. Get Link Status Command Response Data Structure [continued]

Name	Bytes.Bits	Value	Remarks
Link Fault	2.1:2.4	Link Fault	**Bit 2.1:** Returns 1b if PHY has detected a link fault condition. The fault could be anywhere in the PHY layer and either on transmit or receive local or remote fault. The following bits provide additional information about a link fault condition: **Bit 2.2:** Returns 1b if a transmit link fault condition is detected, 0b otherwise. **Bit 2.3:** Returns 1b if a receive link fault condition is detected, 0b otherwise. **Bit 2.4:** Returns 1b if a remote fault condition detected, 0b otherwise.
External Port Link Status	2.5	Port's Link Status	**Bit 2.5:** Port Link Status Returns 1b if link status = up, or returns 0b if the link status = down.
Media Available	2.6	Media Available	**Bit 2.6:** Media Available Returns 1b if media is available for normal link communication, or returns 0b otherwise. 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 when connecting to an external BASE-T PHY or when connecting to a backplane.
Signal Detect	2.7	Signal Detect	**Bit 2.7:** Signal Detect Returns 1b if a receive signal is detected by the PHY or module, or returns 0b otherwise. 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.
AN Completed	3.0	An Completed	**Bit 3.0:** Auto-Negotiation Completed Returns 1b if auto-negotiation completed successfully, or returns 0b otherwise. This bit is valid only if the currently configured PHY type supports auto-negotiation and auto-negotiation is enabled.
LP AN Ability	3.1	LP AN Ability	**Bit 3.1:** Link Partner Auto-Negotiation 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	**Bit 3.2:** Parallel Detection Fault Returns 1b if the PHY detects parallel detection fault, or returns 0b otherwise. This bit is valid only if the PHY supports auto-negotiation with parallel detection enabled.
FEC Enabled	3.3	FEC Enabled	**Bit 3.3:** FEC Enabled Returns 1b if FEC is enabled on the link, or returns 0b otherwise. 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	**Bit 3.4:** Power Mode Returns the Low power configuration as set by the Set PHY Config (0x0601) command 0b = High Power Mode 1b = Low Power Mode

Table 3-40. Get Link Status Command Response Data Structure [continued]

Name	Bytes.Bits	Value	Remarks
Link Pause Status	3.5:3.6	Link Pause Status	**Bit 3.5:** Returns 1b if Tx link pause is enabled on the link during auto-negotiation, or returns 0b otherwise. **Bit 3.6:** Returns 1b if Rx link pause is enabled on the link during auto-negotiation, or returns 0b otherwise. Link Pause status is reported when the current PHY type supports auto-negotiation. Link pause should be disabled if PFC is enabled on the link. Simultaneous operation of link pause and PFC is not supported.
Qualified Module	3.7	Qualified Module	**Bit 3.7:** Qualified Module When the E810 is connected to an external SFP+/QSFP+ module and module 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.
PHY Temp Alarm	4.0	PHY Temp Alarm	**Bit 4.0:** PHT 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	**Bit 4.1:** Excessive Link Errors Returns 1b if an excessive errors over the link condition is reported by the outermost PHY, or returns 0b otherwise
Port Tx Suspended	4.2-4.3	Port Tx Suspended	**Bits 4.2-4.3:** 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	4.4-4.7	Reserved	Reserved.
Loopback Enabled Status	5	Loopback Enabled Status	**Bit 5.0:** PHY local loopback enabled. **Bit 5.1:** PHY remote loopback enabled. **Bit 5.2:** MAC local loopback enabled. **Bit 5.3-5.5:** The PHY index of the PHY that is reported to perform loopback. **All other bits** = Reserved. This command reports the first PHY in the chain that perform loopback, starting at the outermost.
Max Frame Size	6-7	Max Frame Size	Maximum frame size set on this port.
KR FEC Enabled	8.0	KR FEC Enabled	**Bit 8.0:** KR FEC Enabled Returns 1 if KR-FEC was negotiated on the link, or returns 0 otherwise. This field is reported when the current PHY type supports auto-negotiation.
RS 528 FEC Enabled	8.1	RS 528 FEC Enabled	**Bit 8.1:** RS 528 FEC Enabled Returns 1 if RS-FEC 528 was negotiated on the link, or returns 0 otherwise. This field is reported when the current PHY type supports auto-negotiation.
RS 544 FEC Enabled	8.2	RS 544 FEC Enabled	**Bit 8.2:** RS 544 FEC Enabled Returns1 if RS-FEC 544 was negotiated on the link, or returns 0 otherwise. This field is reported when the current PHY type supports auto-negotiation.

Table 3-40. Get Link Status Command Response Data Structure [continued]

Name	Bytes.Bits	Value	Remarks
Pacing Config	8.3-8.7	Pacing Config	**Bits 8.3-8.6:** 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 Table 3-18. **Bit 8.7:** Specifies if the pacing is average rate or fixed IPG.
External Device Power Ability	9.0-9.1	External Device Power Class	**Bits 9.0-9.1:** External Device Power Class QSFP+: 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 is disabled. BASE-T (bit 9.0): 0b = Low and high power. 1b = High power only.
Reserved	9.2-9.7	Reserved	Reserved.
Current Link Speed	10-11	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. **Bit 10.0:** 10 Mb/s **Bit 10.1:** 100 Mb/s **Bit 10.2:** 1000 Mb/s **Bit 10.4:** 5 Gb/s **Bit 10.5:** 10 Gb/s **Bit 10.6:** 20 Gb/s **Bit 10.7:** 25 Gb/s **Bit 11.1:** 50 Gb/s **Bit 11.2:** 100 Gb/s **Bit 11.3:** 200 Gb/s **All other bits** = Reserved, must be zero. This parameter is used by the device 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 firmware or software 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.
Reserved	12-15	Reserved	Must b 0x0.
Current PHY type	16-31	PHY Type	This field will have 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 bytes' encoding is identical to PHY type (PHY capabilities) found in Section 3.3.3.2.1. Additionally, this field can set bit 127 in the structure to indicate that the link state is disabled, or bit 126 in the structure to indicate that PHY is in the auto-negotiation state.

3.2.4.1.6 Link Status Event (0x0607)

The Link Status Event (LSE) is generated by firmware to the device driver when there is a change in status in any of the event causing conditions. Event causing conditions listed in Table 3-41 can be individually masked from generating LSE by using the Set Event Mask (0x0613) command. The LSE share the same opcode, command structure and link status response data structure listed in Table 3-39 and Table 3-40. Firmware posts this data structure to the admin receive queue.

The LSE is disabled by default, unless explicitly enabled by software. Software enables an LSE by setting the *LSE Enable* bit when issuing Get Link Status (0x0607) command (see Table 3-38). Firmware disables the LSE immediately after generating an LSE and does not queue further events until LSE is explicitly enabled by software by the Get Link Status command. Firmware also indicates the LSE enabled status through the *LSE Enable* bit in the Get Link Status command response data structure (see Table 3-39).

The LSE is only generated by firmware to respective PF drivers, and it is software's responsibility to communicate relevant link status change events to the VF through appropriate PF-to-VF communication mechanisms. Software is not expected to use any hardware link status interrupt mechanisms. Hardware link status change interrupts are provided only for diagnostic use. Hence, hardware link status interrupts to PFs and VFs should be disabled for normal operation. Software should use the AQ mechanism to get the link status change notifications using the Get Link Status command and LSE.

Table 3-41. Reported Link Events

Event	Description
Link Change	Link state change. For example, the link state changes from link up to link down.
Media Not Available	Event is reported when an external module is pulled out of its cage.
Link Fault	
PHY Temperature Alarm	Event is generated when an external PHY or module generates a temperature alarm interrupt.
Excessive Errors	
Signal Detect Condition	
Auto-Negotiation Completed	
Module Qualification Failure	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 the link initialization process and then generates this event.
Port Tx Suspend	Indicates that the port's Tx data path is temporarily suspended for configuration purposes.
Topology Conflict	Indicates a conflict in the topology netlist, when the port options of the mezzanine card do not match any available port option of the innermost PHY.
Media Conflict	Indicate a conflict in the media options, when the available options that are supported by the media do not intersect with the port options of the innermost to the outermost PHYs.

3.2.4.1.7 Set Event Mask (0x0613)

This command is used by the device driver to mask the event causing conditions of the link status event from firmware. The link status event is generated by firmware to the PF as described in Section 3.2.4.1.6. This is a Direct command.

Table 3-42. Set Event Mask Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0613	Command opcode.
Datalen	4-5	0x0	Must be 0x0. Value is ignored.
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 firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Logical Port Number	16		Logical Port number. This field specifies the logical port number, and is ignored in the E810.
Reserved	17-23	Reserved	Value 0x0.
Event Mask	24-25	Event Mask	Masks the cause of LSE. 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. **Bit 24.0:** Reserved. **Bit 24.1:** Mask link up/down condition. **Bit 24.2:** Mask media not available or module not present condition. **Bit 24.3:** Mask link fault condition. **Bit 24.4:** Mask PHY temperature alarm condition. **Bit 24.5:** Mask excessive errors over the link condition. **Bit 24.6:** Mask signal detect (asserted or de-asserted) condition. **Bit 24.7:** Mask auto-negotiation completed condition. **Bit 25.0:** Mask module qualification failure condition. **Bit 25.1:** Mask port Tx suspend. **Bit 25.2:** Mask Topology conflict. **Bit 25.3:** Mask media conflict. **All other bits** = Reserved, Must be zero.
Reserved	26-31	Reserved	Must be 0x0. Value is ignored.

The following structure describes the response by firmware to the Set Event Mask command.

Table 3-43. Set Event Mask Command Response

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0613	Command opcode.
Datalen	4-5	0x0	Must be 0x0. Value is ignored.
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 firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Logical Port Number	16		Logical Port number. This field specifies the logical port number, and is ignored in the E810.
Reserved	17-31	Reserved	Must be 0x0. Value is ignored.

3.3 Link Topology

3.3.1 Overview

In the E810, the Ethernet link management task is performed by firmware, using scripting language to allow flexible programming of each link component. To manage the link, the firmware must know which components are connected to the link, how those components are connected to perform the link, and how each component can be accessed to configure it. The scheme of component connections to perform the link is referred as link topology. The link topology might include internal PHYs (within an SoC), external on-board PHY and re-timers, and additional optical or electrical modules and cables either mounted on board or plugged into a cage (for example, SFP+ or QSFP cages). Firmware must know the type of each device connected to the Ethernet link, the sequence of connections between the devices, how to access each device registers, and how to configure it. Internal PHYs might be connected using the SoC internal bus, while external devices are usually connected using the I^2C bus or MDIO, optionally using I^2C multiplexers at the I^2C path. In addition, some control signals are needed for those devices, (for example, RST_N, PRSNT_N and INTR_N). Those control signals are either connected directly to the SoC SDP, or indirectly using an I^2C port replicator connected at the I^2C tree. Finally, the link management task also includes control of link-related LEDs. Firmware must know how to access the LEDs associated with each Ethernet link. Those LEDs might be connected directly to the SoC or indirectly using an I^2C LED driver.

The way that the internal or external components are connected to the Ethernet link is referred to as the link topology. The topology describes how components connected to the Ethernet link and how to access them for configuration. The topology also includes a description of the low-speed devices, such as I^2C multiplexers and port expanders. Link management tasks must configure those I^2C multiplexers to allow access to link components. In addition, link management task must know the port expander connectivity to access the reset and interrupt signals of the link components, and to know if those components exist on the board.

The topology of the main board is stored within an NVM section. During initialization, firmware should read the topology and initialize all devices accordingly. However, some parts of the topology might change dynamically. For example, if an SFP module is inserted into an SFP cage, firmware must detect this event and configure the link appropriately. In addition, the main board might include connectors to PHY mezzanine cards. The firmware must to detect the card, read its topology, and re-build the topology structure used for accessing the link components. The part of the topology that describes the mezzanine card topology is stored on an EEPROM located on the mezzanine card, and the firmware should know how to access it.

Note: The E810 does not support topologies with mezzanine cards.

This section describes the structure of such topology descriptions and how firmware reads and process the topology.

- Section 3.3.2 defines link topology and illustrates the framework of link topology.
- Section 3.3.3 describes the structures needed for link topology in the platform NVM and the ID EEPROM of each mezzanine card. A high level overview of all link structures including the link topology netlist structures is given.
- Section 3.3.4 discuss two examples for the link topology netlist, for main board topology and mezzanine card topology.
- Section 3.3.6 gives detailed information about the link topology netlist structures.
- Section 3.3.7 discuss sizes, conventions, and constraints of the netlist.
- Section 3.3.8 describes the link topology software interface using admin commands.

3.3.2 Link Topology Definition

The link topology describes any device connected to the Ethernet link (for example, PHY or re-timer). The topology also describes the associated devices used for managing the link devices (for example, I^2C multiplexer). In addition, the associated SDPs and LEDs that are used to indicate link changes are defined within the topology description. Those devices might be located on the motherboard (LAN on Motherboard - LOM) or on a mezzanine card, which is plugged-in into the motherboard. Some links might use LOM topology, while others use mezzanine card topology.

Figure 3-15 illustrates link topology.

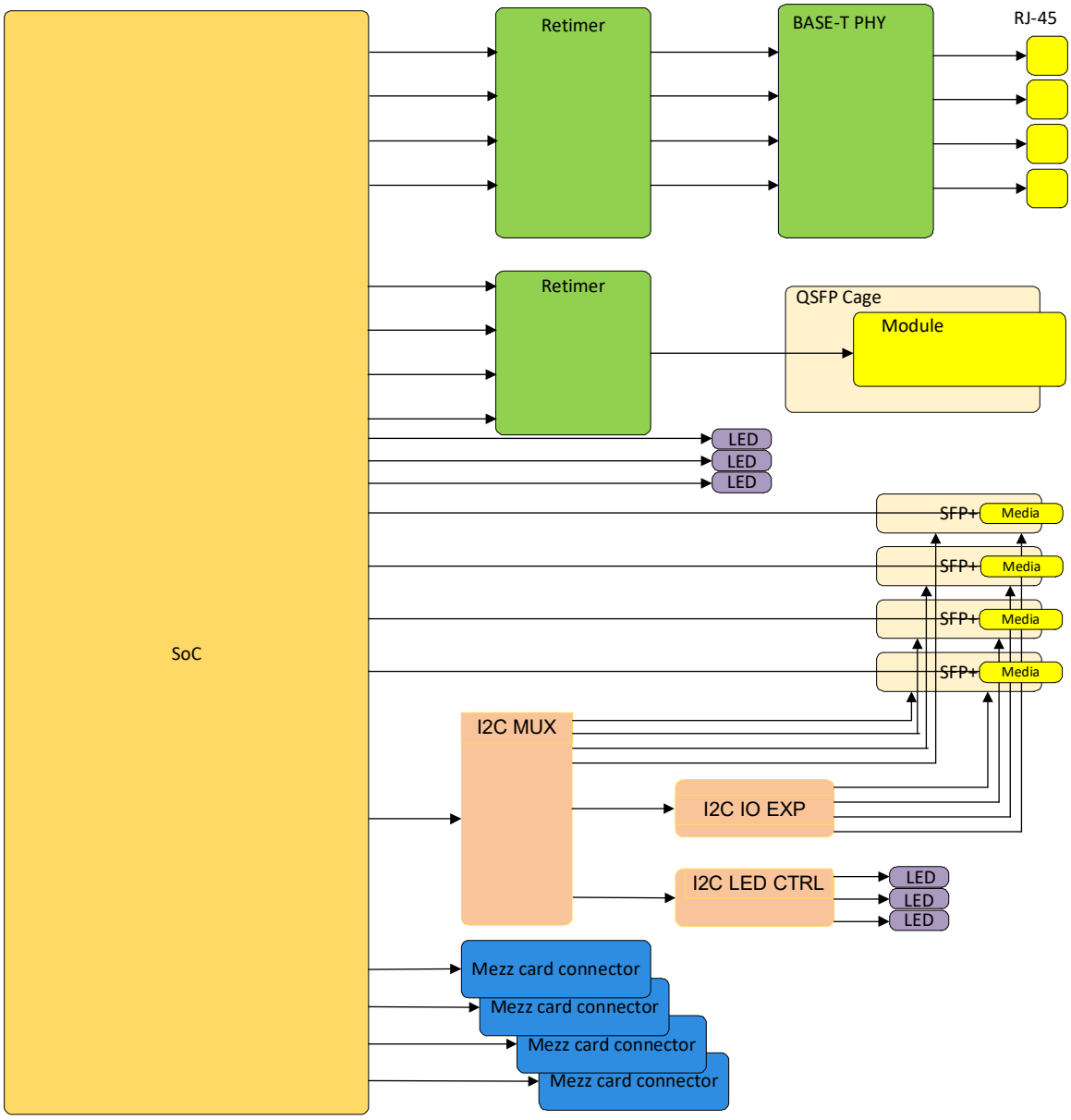


Figure 3-15. Topology Schematic

3.3.3 Topology Structures

The link topology is described using the link topology netlist structure, in both the platform NVM and the mezzanine card ID EEPROM. The E810 registers that are affected by the link topology are modified using the adaptive NVM tables that are stored in the platform NVM.

The link topology netlist describes all topology components and the connection(s) between the, and is composed of several sections. The first on is stored in the link topology NVM section, and describes the motherboard topology. The other topology netlist sections are located in the ID EEPROM of the mezzanine cards, and describe the topology of the mezzanine card. Each mezzanine card ID EEPROM would contain the topology netlist that describes its components.

Some topology-related parameters, referred as link topology capabilities registers, are stored directly as NVM words, and therefore can be modified using adaptive NVM configuration tables. For example, the number of available ports and link speeds is determined by the link topology netlist, and is applied to E810 registers using adaptive NVM. Those parameters are therefore located in the NVM and are subject to adaptive NVM configuration.

The link topology netlist and link topology capabilities sections are also used by the link establishment state machine to bring up the link with the link partner. The NVM stores the adaptive NVM features table, which describes the link topology capabilities registers values and other NVM values, per Ethernet link for a list of predefined link modes.

A mezzanine card ID EEPROM would hold a the link mode configuration for each of the links implemented on the mezzanine card. The link modes configurations for the Ethernet links implemented on the motherboard (that is, LOM links) are also held in the platform NVM.

The topology netlists that are stored at the platform NVM and at the ID EEPROM device of each mezzanine card are merged into one netlist representing the current topology netlist. The NVM section stores the details about the topology components that are on the motherboard, while the ID EEPROM sections stores the details about each mezzanine card topology. A link topology that does not include mezzanine cards, would only have one topology section in the NVM. For a board that is based on mezzanine cards, all topology structures, one from the NVM and one from each mezzanine card, should be merged into the board topology netlist.

This topology structure holds the functionality of the device, the connectivity description of its management interfaces, the way to access its registers and additional information depending on its function. For example, a PHY device would also hold a table that describes how to configure its host-side link, based on the line-side link parameters that were agreed.

At initialization stage, when a module is plugged into a cage and when a mezzanine card is detected, the firmware reads the different topology structures and builds the current topology of the links.

3.3.3.1 High-Speed PHY Chain

One part of the link topology can be viewed as a chain of PHYs, re-timers, and modules that are connected in a chain to perform a link. For the sake of simplicity, we treat all the elements in the chain as PHYs, with two ports: the line port, and the host port, where the line port is the port connected to the chain ended in the line side of the link, and the host port is the port connected to the chain ended at the host.

Figure 3-16 illustrates the PHY chain.

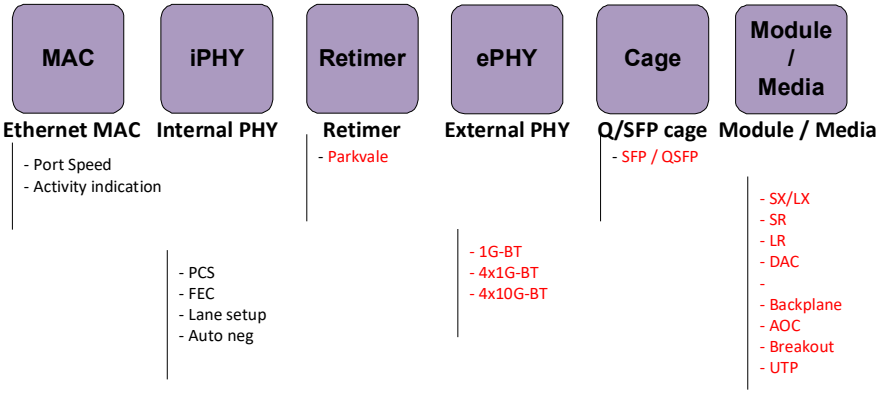


Figure 3-16. High-Speed PHY Chain

Notes:

- Some components in the PHY chain are within the SoC, while others are on the motherboard or mezzanine card.
- Some topologies do not include all components (for example, 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, or a QSFP cage might be plugged with breakout cable, that creates four different Ethernet links.

3.3.3.2 PHY Capabilities Structures

Ethernet link might work in different speeds and different electrical parameters. In general, each speed is specified with several different electrical specifications, to support different media. Such are short-reach cables (SR), long-reach cables (LR), different types of backplanes, and BASE-T pairs. We use single structure to describe the available modes, while each mode might have additional options that are described in adjacent parameters. The PHY capabilities structure, which is also referred to as PHY_TYPE, is used in several topology structures, admin command parameters, and in PHY configuration scripts as a parameter. For the topology description, it is used as a link topology NVM word, that describes the outermost PHY capabilities. It is also used twice, for each PHY node in the topology structures, to indicate the supported modes at the line side and at the host side.

Some capabilities are valid only for the outermost PHY, while others can be reflected also at the board connection between two PHYs in a chain. Table 3-44 and Table 3-45 describe the PHY capabilities for outermost PHYs, which covers all types of links.

Note: For the sake of readability, these tables are organized with capabilities bit per speed.

To avoid explosion of the topology netlist, we implement two different structures. The extended structure is used for the line side of the outermost PHY, which includes all capabilities. The basic structure used for the host side of the outermost PHY and both line side and host side for the other PHYs in the chain.

The extended structure is implemented as 128-bit words (84 bits used and the rest reserved for future use) and is described in Table 3-46. The basic structure is implemented as 64-bit words (31 bits used and the rest reserved for future use) and is described in Table 3-47. Both PHY capabilities structures are implemented in the topology netlist PHY capabilities section, as described in Section 3.3.6.6.

128 bits for outermost line side. 64 bits for everything else.

Table 3-44. PHY Capabilities Bit Matrix (Part 1)

Speed/Media Type	100M	1G	2.5G	5G	10G
BASE-T	100BASE-TX	1000BASE-T	2.5GBASE-T	5GBASE-T	10GBASE-T
Short Reach	1000BASE-SX[1]	1000BASE-SX			10GBASE-SR
Long Reach	1000BASE-LX[2]	1000BASE-LX			10GBASE-LR
Back Plane		1000BASE-KX	2.5GBASE-KX	5GBASE-KR	10GBASE-KR
Serial/Direct Attach	100M-SGMII	1G-SGMII	2.5GBASE-X		10G-SFI-DA
Active Optical/Copper Cable					10G-SFI-ACC / 10G-SFI-AOC
Chip to Chip					10G-SFI-C2C

1. 1000BASE-SX running at speed of 100 Mb/s
2. 1000BASE-LX running at speed of 100 Mb/s

Table 3-45. PHY Capabilities Bit Matrix (Part 2)

Speed/ Media Type	25G	40G	50G	100G	200G	400G
BASE-T	25GBASE-T					
Short Reach	25GBASE-SR	40GBASE-SR4		100GBASE-SR4	200GBASE-SR4	
Long Reach	25GBASE-LR	40GBASE-LR4		100GBASE-LR4	200GBASE-LR4	400GBASE-LR8
Back Plane	25GBASE-KR	40GBASE-KR4	50GBASE-KR2	100GBASE-KR4	200GBASE-KR4-PAM4	
KR-S	25GBASE-KR-S					
AUI	25G-AUI-C2C	40G-XLAUI	50GBASE-LAUI2	100G-CAUI4	200G-AUI4	400G-AUI8
Active Optical/ Copper Cable	25G-AUI-ACC / 25G-AUI-AOC	40G-XLAUI-ACC / 40G-XLAUI-AOC	50G-LAUI2-AOC / 50G-LAUI2-ACC	100G-CAUI4-ACC / 100G-CAUI4-AOC	200G-AUI4-ACC / 200G-AUI4-AOC	400G-AUI8-ACC / 400G-AUI8-AOC
CR	25GBASE-CR	40GBASE-CR4	50GBASE-CR2	100GBASE-CR4		
CR-S	25GBASE-CR-S					
CP			50GBASE-CR-PAM4	100GBASE-CR2-PAM4	200GBASE-CR4-PAM4	
	25GBASE-CR1		50GBASE-KR-PAM4	100GBASE-KR2-PAM4		
			50GBASE-LR	100GBASE-DR	200GBASE-DR4	400GBASE-DR4
			50GBASE-SR	100GBASE-SR2		
			50GBASE-FR		200GBASE-FR4	400GBASE-FR8
	25GBASE-KR1		50GBASE-AUI1	100G-AUI4	200G-AUI8	
			50G-AUI1-AOC / 50G-AUI1-ACC	100G-AUI4-AOC / 100G-AUI4-ACC	200G-AUI8-ACC / 200G-AUI8-AOC	
			50GBASE-AUI2	100G-AUI2		
			50G-AUI2-AOC/ 50G-AUI2-ACC	100G-AUI2-AOC / 100G-AUI2-ACC		
				100GBASE-CR-PAM4		
				100GBASE-KR-PAM4		

Note: As NVM word, the capabilities structure describes the enabled options for the outermost PHY. These link speeds and link options might or might not be supported at the specified device level, or for specific link. The enabled speeds/options in Table 3-44 and Table 3-45 should be selected in such a way that the device would be able to support them. For example, the E810 does not support 2.5G/40G/200G/400G speeds, and therefore those options should not be selected for this device.

3.3.3.2.1 Extended PHY Capabilities 128-Bit Word Structure

Table 3-46. 128-Bit Word Extended PHY Capabilities

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	Reserved	25G-AUI-C2C	25G-AUI-AOC/ACC	25GBASE-KR1	25GBASE-KR-S	25GBASE-KR	25GBASE-LR	25GBASE-SR	25GBASE-CR1	25GBASE-CR-S	25GBASE-CR	25GBASE-T	10G-SFI-C2C	10G-SFI-AOC/ACC	10GBASE-KR	10GBASE-LR	10GBASE-SR	10G-SFI-DA	10GBASE-T	5GBASE-KR	5GBASE-T	Reserved	Reserved	Reserved	1G-SGMII	1000BASE-KX	1000BASE-LX	1000BASE-SX	1000BASE-T	100M-SGMII	100BASE-TX
100GBASE-DR	100GBASE-SR2	100GBASE-CR2-PAM4	100GBASE-KR4-PAM4	Reserved	100G-AUI4	100G-AUI4-AOC/ACC	100G-CAUI4	100G-CAUI4-AOC/ACC	100GBASE-KR4	100GBASE-LR4	100GBASE-SR4	100GBASE-CR4	50G-AUI1	50G-AUI1-AOC/ACC	50GBASE-KR-PAM4	50GBASE-LR	50GBASE-FR	50GBASE-SR	50GBASE-CR-PAM4	50G-AUI2	50G-AUI2-AOC/ACC	50G-LAUI2	50G-LAUI2-AOC/ACC	50GBASE-KR2	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Reserved												400G-AUI8	400G-AUI8-AOC/ACC	400GBASE-DR4	400GBASE-LR8	400GBASE-FR8	200G-AUI8	200G-AUI8-AOC/ACC	200G-AUI4	200G-AUI4-AOC/ACC	200GBASE-KR4-PAM4	200GBASE-DR4	200GBASE-LR4	200GBASE-FR4	200GBASE-SR4	200GBASE-CR4-PAM4	100G-AUI2	100G-AUI2-AOC/ACC	Reserved	Reserved	100GBASE-KR2-PAM4
Reserved																															

3.3.3.2.2 Basic PHY Capabilities 64-Bit Word Structure

Table 3-47. 64-Bit Word Extended PHY Capabilities

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	21	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	400G-AUI8	200G-AUI8	200G-AUI4	200GBASE-KR4	100G-AUI2	Reserved	100GBASE-KR-PAM42	100GBASE-KP4-PAM4	100G-AUI4	100G-CAUI4	100GBASE-KR4	50G-AUI1	50GBASE-KR-PAM4	50G-AUI2	50G-LAUI2	50GBASE-KR2	Reserved	Reserved	25G-AUI-C2C	25GBASE-KR1	25GBASE-KR-S	25GBASE-KR	10G-SFI-C2C	10GBASE-KR/CR1	10G-SFI-DA	5GBASE-KR	Reserved	Reserved	1G-SGMII	1000BASE-KX	100M-SGMII
Reserved																															

3.3.3.3 Link Modes Adaptive NVM Features Tables

The link topology applied by the link topology netlist might affect can E810 CSRs. This is done using adaptive NVM method. The link topology netlist references two types of adaptive NVM features tables.:

- **Global super-feature tables** — Affect E810 configurations that should be modified according to the combination of the valid ports and their corresponding maximal speed.
- **Per-port/Per function features tables** — Affect configurations relevant for the specific port type and speed.

The global super-feature table is applied by the adaptive NVM configuration ID at the innermost PHY attached to the selected port option. There should be a different port option for each valid combination of E810 ports, as described in Table 3-12, "Port Configurations".

Table 3-48 describes the main E810 settings that are affected by the global super-feature.

Table 3-48. Global Super-Feature Affected Settings

Feature	Description
E810 port topology mode	Controls the allocation of E810 resources according to the number of ports and speed of those ports.
Additional DCB configurations	DCB-to-UP translation tables. Congestion domain configurations.
VSI/VEB allocation	Allocations of resources to physical functions (see Section 7.8.10).
Max number of virtual functions allocated to physical function	
Physical function port association	Association of physical function to port (see Section 4.5).

Table 3-49 describes the main E810 settings that are affected by the Per-port feature.

Table 3-49. Per-Port Super Feature Affected Settings

Feature	Description
Port enabling	Enable or disable the port (see Section 4.5).
Port WoL setting	Enable/Disable this port for WoL setting.

Table 3-50 describes the main E810 settings that are affected by the Per-function feature.

Table 3-50. Per-Function Super Feature Affected Settings

Feature	Description
Physical function enabling	Enabling the physical function that is associated with the port (see Section 4.5).
Device ID	

3.3.3.4 Link Topology Netlist

The topology netlist is a description of all components 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 topology device (node), with a description of its interfaces, capabilities, device types, associated configuration, and any information needed to setup this device to bring up the Ethernet link.

Each topology node is composed of several sections. Two sections are common for all topology nodes and are reflected the topology netlist:

- **Node Header Section** — Includes the basic description of the node and pointers to the other sections.
- **Node I/O Section** Includes pointers and descriptions of the node I/Os that connect the node toward the host.

Some topology devices do not need any additional description besides the Node 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 need and additional scratch section, which is used by the scripts to store scratch data.

The following basic topology devices contain only the Node Header Section and the Node I/O Section, with some of the nodes also containing also the scratch area (Node Scratch Section):

- GPIO controller node
- MUX controller node
- LED controller node
- ID EEPROM node
- Cage node
- Mezzanine connector node

The Node Header Section and the Node I/o Section that compose the basic topology device structure include the following information:

- **Node Header Section** — Contains information on the following:
  - — Node type, node handle, node address, node part number, and 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.
- **Node I/O Section** — Includes a list of interfaces that are connected to this node. As all topology devices are eventually controlled by the SoC, the link topology netlist is built as a tree, where the roots of the tree are the SoC nodes that represent I/Os that are directly controlled by the SoC firmware. Therefore, it is sufficient that each topology node would list only the interfaces (I/Os) that connect between the node and other nodes that are up in the topology tree. Those I/Os could be either I/Os that are controlled by this node (master I/Os), or I/Os that are used to control this node (slave I/Os). 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 (for example, I^2C, MDIO, SDP, LED), and the function of the interface (for example, when is the type SDP, what is the functionality of that SDP - INT_N / PRESET_N / TX_DISABLE, and so on).

— 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.

For some devices, like the LEDs and the thermal sensors, we need to add an information about the association of the topology device with another topology device, referred as its parent device. The port option selected for the parent device is also applied for the current topology device. Therefore, LED and thermal sensor nodes include additional node parent section.

The LED topology node includes the basic node sections, the node parent section and additional LED configuration section for each port option, with the following information:

- LED link affinity (Lo /High) — 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 affects the LED. The highest two bits are used to select the logic operation between the ports (AND or OR).
- LED flag (Low / High)
- LED color and condition

The Temperature Sensor node includes the basic node sections, the node parent section, and additional Thermal Sensor 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.

The PHY node includes several sections that are added to the basic node description. A PHY node would include two sections of PMD analog options (one for the host side and one for the line side) and two sections of PMD miscellaneous analog options (one for the host side and one for the line side). The PMD analog options contain configurations and coefficients for the analog side of the PHY lines, while the PMD miscellaneous analog section might contain additional configuration that is PHY-specific.

In addition to the above sections, the PHY node also contains one port option pointer section and port option header sections per port option. Each port option contains several PMDs, with PHY capability section per PMD within the port option. See Section 3.3.6.13 for an example of a PHY node.

The PMD Analog Section contains the following information:

- Coefficients for 1G/5G/10G/25G/50G AUI modes.
- Coefficients for 10G KR, 25G KR/CR and 50G KR modes.
- Link establishment state machine timeouts for 1G/5G/10G/25G/50G AUI modes.
- Link establishment state machine timeouts for 10G KR, 25G KR/CR and 50G KR modes.
- Lane transmit and receive polarity.

The Port Option Pointer Section includes the following information, per Port option:

- A pointer to the relevant port option section.

The Port Option Header Section includes the following information:

- Adaptive NVM Global Super Configuration ID — A pointer to the adaptive NVM super-feature table that sets NVM parameters per-port option.
- Adaptive NVM PHY Configuration ID — A pointer to the adaptive NVM feature table of the relevant PHY that sets NVM parameters per-port option of the PHY.
- 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:

- Feature IDs and configuration IDs of adaptive NVM tables of the port and the function, relevant for this port option.
- Link options (EEE enable, FEC abilities advertise and request for 25G (cl. 74,91), FEC abilities advertise and request for 10G, PCS enable, Low power mode enable, LPLU enable, Rx and Tx pause, auto-negotiation options (cl. 28, 37,73), LESM enable, Auto FEC enable, host and line lane reversal, and outermost indication).
- PMD width for host side and line side.
- First lane (lane 0) ID for host side and line side.
- EEE TWSysTX timer value.
- PHY capabilities structure for host side and line side per Section 3.3.6.5.5.

Table 3-51 summarize the link topology node structure, for different nodes:

Table 3-51. Link Topology Node Structure

Section	Description	Applies to Node Types	Number of Sections in a Node
Header Section	The header section contains basic information for all node types, such as node type, section length, address and pointers to the other node sections.	All nodes types	One section. Fixed size
I/O Section	The IO 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.
PMD Analog Section	This section contains the configuration of the PHY lanes analog side.	PHY nodes	Two section per PHY node, 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 section 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.

Table 3-51. Link Topology Node Structure [continued]

Section	Description	Applies to Node Types	Number of Sections in a Node
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.

3.3.3.5 Link Topology NVM Section

In addition to the link topology netlist, there is an additional NVM section for link topology parameters that might be updated by the link topology firmware. The following parameters are included in this section.

Table 3-52. Link Topology NVM Section

Parameter	Word/Bit Offset	Description
Netlist Map Version	0x2 / Bits [15:8]	The version of the netlist field definition
Load Mode	0x2 / Bit 0	Netlist Loading Mode — Controls the steps taken at the netlist load time. Valid only for the motherboard's netlist. Two modes defined: 0x0 = Normal mode — the active port options are loaded. 0x1 = resolution mode — The active port options of the non-innermost nodes are 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.
Netlist Version	0x3 / Bits [15:0]	The version of the netlist.
Active Port Option	0x4-0x13 / Bits [3:0] and Bits [11:8]	The active port option selected for each PHY in the netlist. Up to 32 PHYs are supported, where each word holds the active option for two PHYs. The first is in Bits [3:0] and the second is in Bits [11:8].
Forced	0x4-0x13 / Bit 4 and Bit 12	Indicates that the active port option is forced. 0x0 = Not forced — Port option is not forced. The active port option can be chosen based on automatic topology or media conflict resolution. 0x1 = Forced — Port option is forced to the value in the *Active Port Option* field. When the netlist is read from NVM and mezzanine cards, the innermost PHY port option is selected based on the current active port option.

3.3.4 Link Topology Use Cases

This section illustrates the resulting link topology netlist, with a specific use case. We take our use cases from the *NAC Hardware Topology Design Specification* document, focusing on one topology for the main board and one for a mezzanine card. The following use cases are taken:

- A main board (or LAN on motherboard - LOM) use case, where all link topology components are on the motherboard. We focus on "CFG3.x - 1xQSFP + CEI".
- A mezzanine card use case. A typical mezzanine card with PHYs, re-timers, LED controllers, and I/O expander is shown. We focus on "CEI CFG2 - PKVL 4x SFP".

3.3.4.1 LOM Use Case

Figure 3-17 describes typical LAN on motherboard use case. Figure 3-18 describes the resulting topology data structures.

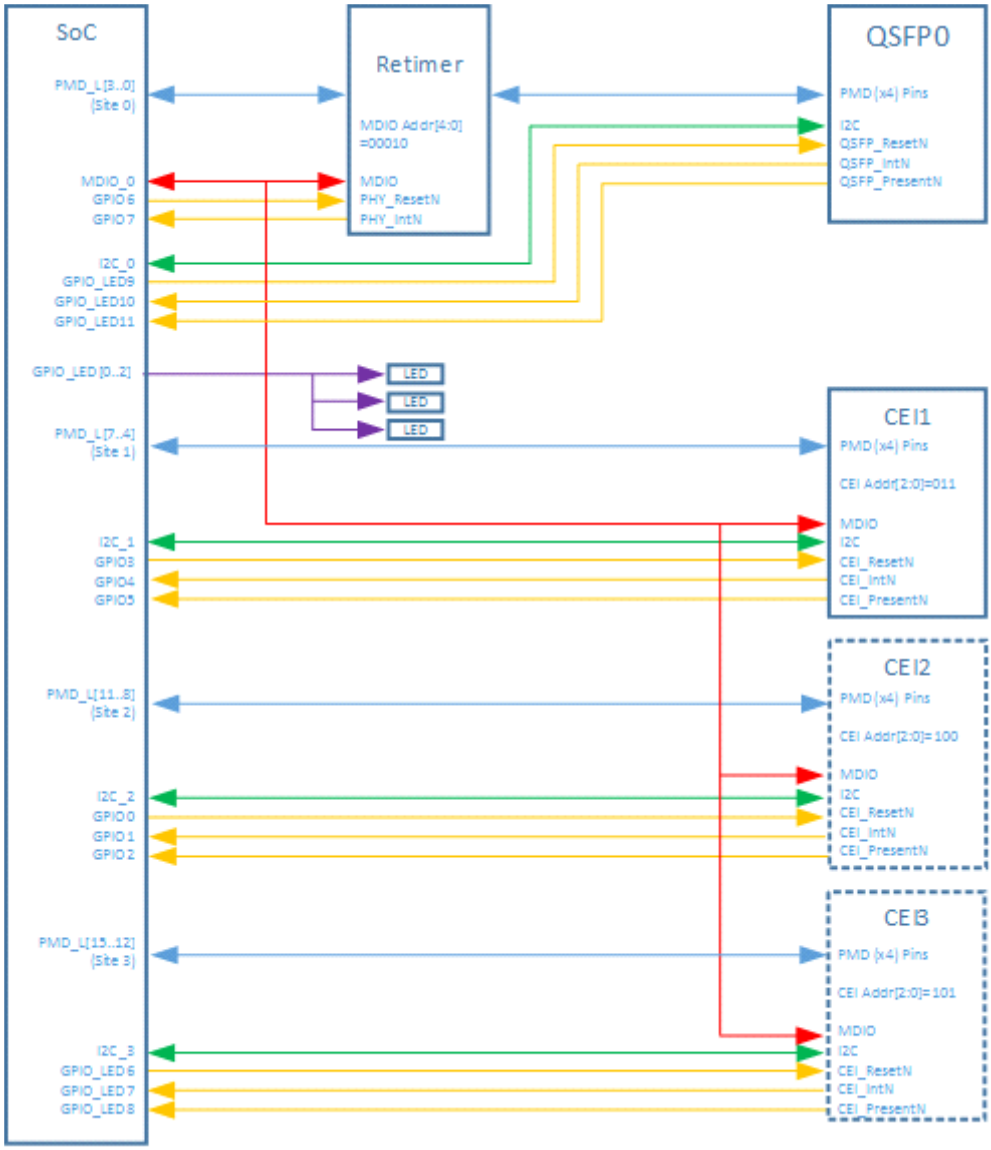


Figure 3-17. LOM Topology Use Case

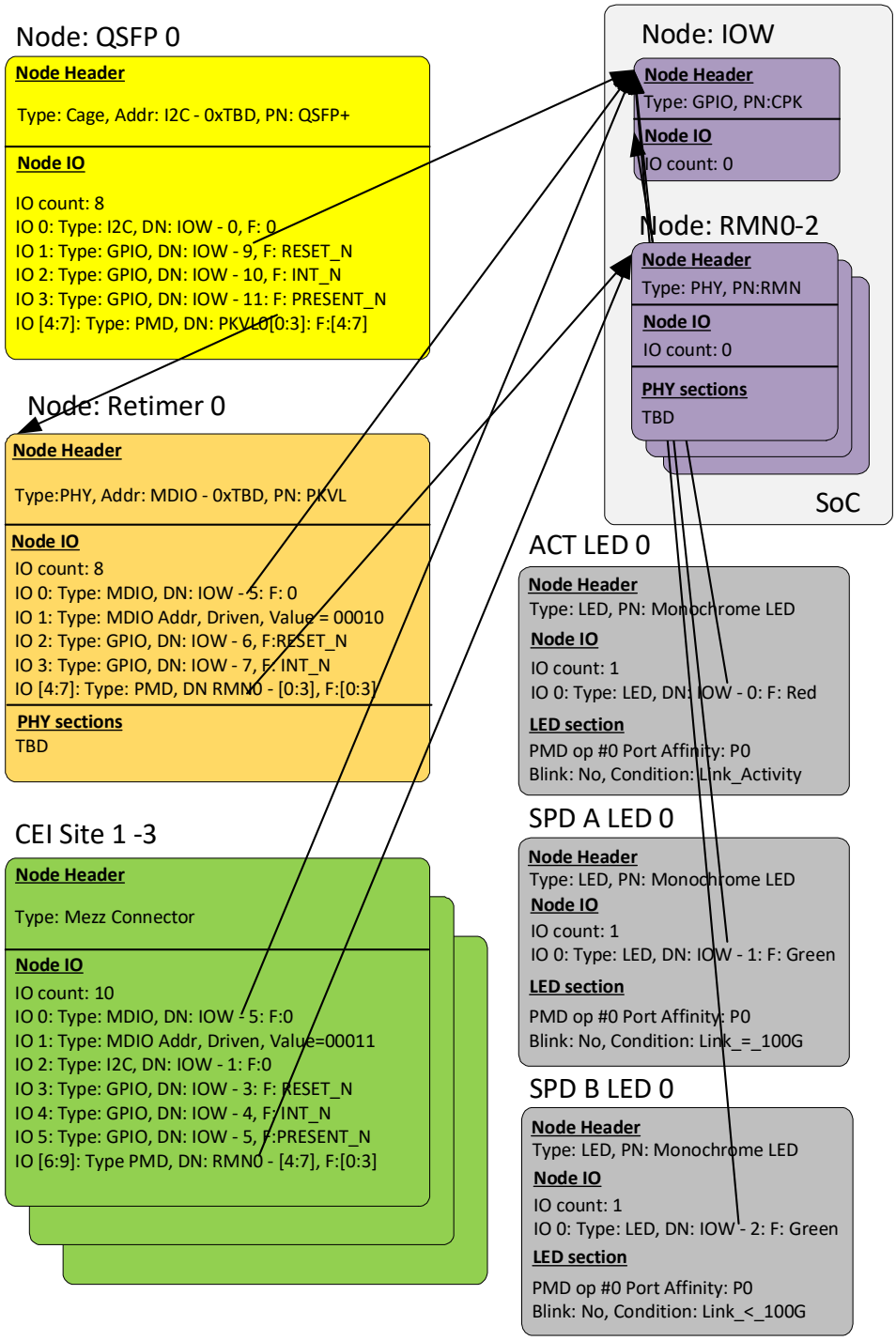


Figure 3-18. LOM Topology Use Case Data Structures

3.3.4.2 Mezzanine Card Use Case

Figure 3-19 describes typical Mezzanine card use case. Figure 3-20 describes the resulting topology data structures.

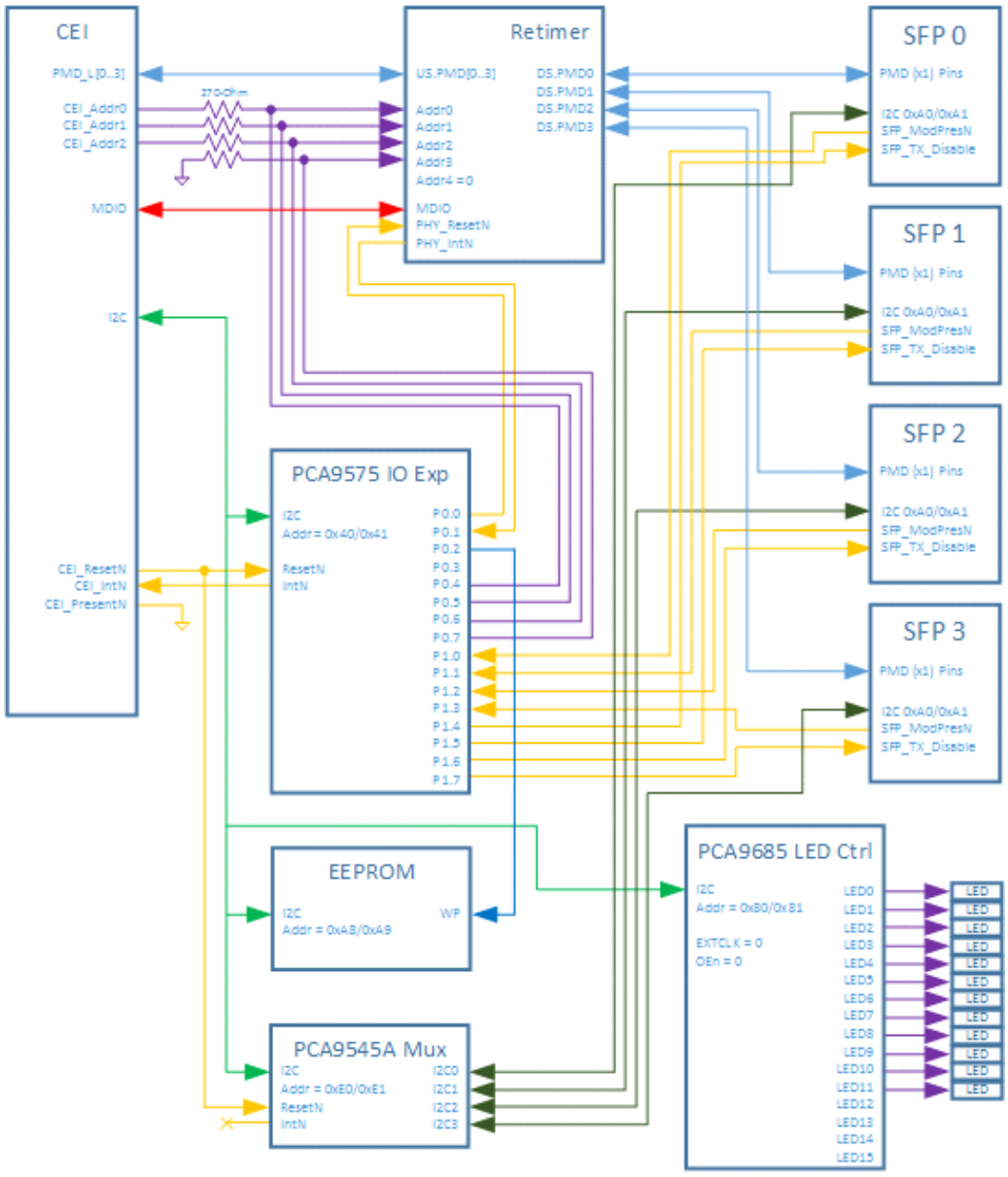


Figure 3-19. Mezzanine Card Topology Use Case

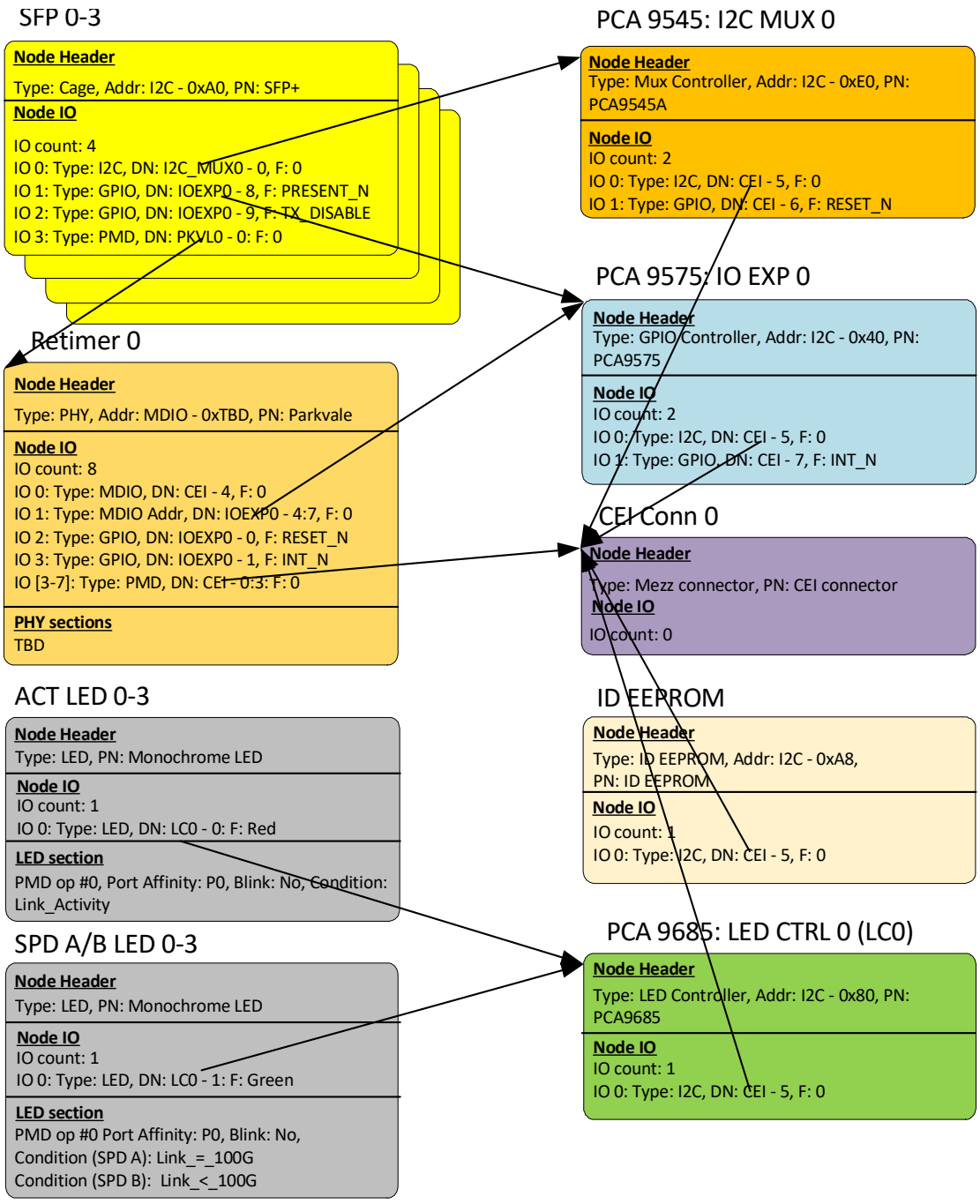


Figure 3-20. Mezzanine Card Topology Use Case Data Structures

3.3.5 Topology Initialization

At boot time, or when an interrupt for card insertion or module insertion is triggered, firmware should check the signature of all mezzanine cards. When a card is detected, it should compare the OCP ID and version against a local copy of the currently-populated mezzanine cards. Any change in the card should trigger a topology structures build.

3.3.6 Topology Netlist

This section defines the structure of the topology netlist module of the NVM and the mezzanine ID EEPROM. The module 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.

Note: This section defines the topology netlist for several products, therefore some entries include descriptions of values that are not supported in the current product. Those values are usually marked in the Magenta color.

3.3.6.1 Topology Netlist Header

This section describes the length of the Topology Netlist Section and the number of nodes it holds.

Word Offset	Description	Section Reference
0x0000	Module Length	3.3.6.1.1
0x0001	Node Count	3.3.6.1.2
0x0002	Netlist Map Version and CRC	3.3.6.1.3
0x0003	Netlist Version	3.3.6.1.4
0x0004 + n*2 (0x0004 for n=0)	Node Handle (for Node 0)	3.3.6.1.5
0x0005 + n*2 (0x0005 for n=0)	Node Block Offset (for Node 0)	3.3.6.1.6
0x0004 + n*2 (0x0006 for n =1)	Node Handle (for Node 1)	3.3.6.1.5
0x0005 + n*2 (0x0007 for n =1)	Node Block Offset (for Node 1)	3.3.6.1.6
...	...	...
0x0004 + (Node Count - 1)*2	Node Handle for last node (n = Node Count -1)	3.3.6.1.5
0x0005 + (Node Count - 1)*2	Node Block Offset for last node (n = Node Count -1)	3.3.6.1.6
0x0004 + Node Count*2	First Word (Offset 0x0) of the Netlist Identifier Block	

Word Offset	Description	Section Reference
…	…	…
0x0033 + Node Count*2	Last Word (0ffset 0x2F) of the Netlist Identifier Block	3.3.6.1.30

Netlist Identifier Block:

Word Offset	Description	Section Reference
0x0000	Module ID - Netlist Identifier Block	3.3.6.1.7
0x0001	Module Length - Netlist Identifier Block	3.3.6.1.8
0x0002	Base Release Version - Major Low Word- Netlist Identifier Block	3.3.6.1.9
0x0003	Base Release Version - Major High Word - Netlist Identifier Block	3.3.6.1.10
0x0004	Base Release Version - Minor Low Word - Netlist Identifier Block	3.3.6.1.11
0x0005	Base Release Version - Minor High Word - Netlist Identifier Block	3.3.6.1.12
0x0006	Base Release Version: Type Low Word - Netlist Identifier Block	3.3.6.1.13
0x0007	Base Release Version: Type High Word - Netlist Identifier Block	3.3.6.1.14
0x0008	Base Release Version: Revision Low Word - Netlist Identifier Block	3.3.6.1.15
0x0009	Base Release Version: Revision High Word - Netlist Identifier Block	3.3.6.1.16
0x000A - 0x0019	Netlist Binary Hash Word - Netlist Identifier Block	3.3.6.1.17
0x001A	Netlist Origin Flags: Low - Netlist Identifier Block	3.3.6.1.18
0x001B	Netlist Origin Flags: High - Netlist Identifier Block	3.3.6.1.19
0x001C	Netlist Modification Date: Year - Netlist Identifier Block	3.3.6.1.20
0x001D	Netlist Modification Date: Day/Month - Netlist Identifier Block	3.3.6.1.21
0x001E	Netlist Modification Date: Time - Netlist Identifier Block	3.3.6.1.22
0x001F	ETT Version Used for Netlist Compile: Major - Netlist Identifier Block	3.3.6.1.23
0x0020	ETT Version Used for Netlist Compile: Minor - Netlist Identifier Block	3.3.6.1.24
0x0021	ETT Version Used for Netlist Compile: Revision - Netlist Identifier Block	3.3.6.1.25
0x0022	ETT Version Used for Netlist Compile: Patch - Netlist Identifier Block	3.3.6.1.26
0x0023 - 0x002C	Full git Netlist SHA-1 Hash Word - Netlist Identifier Block	3.3.6.1.27
0x002D	Customer IANA: Low Word - Netlist Identifier Block	3.3.6.1.28
0x002E	Customer IANA: High Word - Netlist Identifier Block	3.3.6.1.29
0x002F	Customer Netlist Version Word - Netlist Identifier Block	3.3.6.1.30

The section is not accessible through the GETNODEATTR action.

3.3.6.1.1 Module Length (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		Module length in double words. Does not include the first DWord that contains the *Module Length* field itself and the N*ode Count* field.

3.3.6.1.2 Node Count (0x0001)

Bits	Field Name	Default NVM Value	Description
15:10	Reserved		Reserved.
9:0	Node Count		The number of nodes captured in the topology netlist.

3.3.6.1.3 Netlist Map Version and CRC (0x0002)

Bits	Field Name	Default NVM Value	Description
15:8	Netlist Map Version	0x01	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

3.3.6.1.4 Netlist Version (0x0003)

Bits	Field Name	Default NVM Value	Description
15:0	Netlist Version	1	The version of the netlist. This field changes when the content of a tracked netlist changes. That is, the values of some of the fields in the netlist change.

3.3.6.1.5 Node Handle (0x0004 + n*2)

Bits	Field Name	Default NVM Value	Description
15:10	Reserved		Reserved.
9:0	Node Handle		Unique identifier of the node in the topology. Matches the *Node Handle* field inside the corresponding Node Header Section. See Section 3.3.6.2.2.

3.3.6.1.6 Node Block Offset (0x0005 + n*2)

Bits	Field Name	Default NVM Value	Description
15:0	Node Block Offset		The word offset of the node section relative to the beginning of the module.

Note: The next nodes are referenced with Net-list Identifier Block offset, which is equal to Node Block Offset + 0x0004 + NodeCount * 2.

3.3.6.1.7 Module ID - Netlist Identifier Block + 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Module ID	0x0000	Module ID.

3.3.6.1.8 Module Length - Netlist Identifier Block + 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Module Length	0x0000	Module length in double words. Does not include the *Module Length* field.

3.3.6.1.9 Base Release Version: Major Low Word - Netlist Identifier Block + 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	Base Release Version: Major Low Word	0x0000	Low word of the Base Release version: Major value. Stored in BCD (binary coded decimal). Similar to NVM version.

3.3.6.1.10 Base Release Version: Major High Word - Netlist Identifier Block + 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	Base Release Version: Major High Word	0x0000	High word of the Base Release version: Major value. Stored in BCD (binary coded decimal). Similar to NVM version.

3.3.6.1.11 Base Release Version: Minor Low Word - Netlist Identifier Block + 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	Base Release Version: Minor Low Word	0x0000	Low word of the Base Release version: Minor value. Stored in BCD (binary coded decimal). Similar to NVM version.

3.3.6.1.12 Base Release Version: Minor High Word - Netlist Identifier Block + 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	Base Release Version: Minor High Word	0x0000	High word of the Base Release version: Minor value. Stored in BCD (binary coded decimal). Similar to NVM version.

3.3.6.1.13 Base Release Version: Type Low Word - Netlist Identifier Block + 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	Base Release Version: Type Low Word	0x0000	Low word of the Base Release version: Type value. Each unique netlist from Intel has a different Type value. Stored in BCD (binary coded decimal). Similar to NVM version.

3.3.6.1.14 Base Release Version: Type High Word - Netlist Identifier Block + 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	Base Release Version: Type High Word	0x0000	High word of the Base Release version: Type value. Each unique netlist from Intel will have a different Type value. Stored in BCD (binary coded decimal). Similar to NVM version.

3.3.6.1.15 Base Release Version: Revision Low Word - Netlist Identifier Block + 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	Base Release Version: Revision Low Word	0x0000	Low word of the Base Release revision. This is updated by ETT on any change to the template. Automatic by ETT.

3.3.6.1.16 Base Release Version: Revision High Word - Netlist Identifier Block + 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	Base Release Version: Revision High Word	0x0000	High word of the Base Release revision. This is updated by ETT on any change to the template. Automatic by ETT.

3.3.6.1.17 Netlist Binary Hash Word: Netlist Identifier Block + 0x000A-0x0019

Bits	Field Name	Default NVM Value	Description
15:0	Hash Value of Netlist	0x0000	Hash of Netlist complied binary. Inserted by ETT. It should be the SHA3 256 bit hash of the netlist where the Hash Field is set to 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.

3.3.6.1.18 Netlist Origin Flags Low: Netlist Identifier Block + 0x001A

Bits	Field Name	Default NVM 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.

3.3.6.1.19 Netlist Origin Flags High: Netlist Identifier Block + 0x001B

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0000	Reserved.

3.3.6.1.20 Netlist Modification Date: Year - Netlist Identifier Block + 0x001C

Bits	Field Name	Default NVM Value	Description
15:0	Netlist Modification Date: Year	0x0000	Year of the last netlist modification in BCD (binary coded decimal). Set by ETT on any Intel or customer modification of the netlist. Example: 0x2019 = 2019.

3.3.6.1.21 Netlist Modification Date: Day/Month - Netlist Identifier Block + 0x001D

Bits	Field Name	Default NVM Value	Description
15:8	Netlist Modification Date: Day	0x00	Day of the month of last netlist modification in BCD (binary coded decimal). 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 BCD (binary coded decimal). Set by ETT on any Intel or customer modification of the netlist. Example: 0x11 = the 11th months, which is November.

3.3.6.1.22 Netlist Modification Date: Time - Netlist Identifier Block + 0x001E

Bits	Field Name	Default NVM Value	Description
15:0	Netlist Modification Date: Time	0x0000	Time of the last netlist modification in BCD (binary coded decimal). Set by ETT on any Intel or customer modification of the netlist. Example: 0x1300 is 1:00 pm.

3.3.6.1.23 ETT Version Used for Netlist Compile: Major - Netlist Identifier Block + 0x001F

Bits	Field Name	Default NVM Value	Description
15:0	ETT Major Version Used for Netlist Compile	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.

3.3.6.1.24 ETT Version Used for Netlist Compile: Minor - Netlist Identifier Block + 0x0020

Bits	Field Name	Default NVM Value	Description
15:0	ETT Minor Version Used for Netlist Compile	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.

3.3.6.1.25 ETT Version Used for Netlist Compile: Revision - Netlist Identifier Block + 0x0021

Bits	Field Name	Default NVM Value	Description
15:0	ETT Revision Version Used for Netlist Compile	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.

3.3.6.1.26 ETT Version Used for Netlist Compile: Patch - Netlist Identifier Block +0x0022

Bits	Field Name	Default NVM Value	Description
15:0	ETT Patch Version Used for Netlist Compile	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.

3.3.6.1.27 Full git Netlist SHA-1 Hash Word - Netlist Identifier Block + 0x0023-0x002C

Bits	Field Name	Default NVM 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.

3.3.6.1.28 Customer IANA: Low Word - Netlist Identifier Block + 0x002D

Bits	Field Name	Default NVM Value	Description
15:0	Customer IANA - Low Word	0x0157	Customer IANA low. Intel to set Intel IANA 0x0157. Customer to fill in their IANA.

3.3.6.1.29 Customer IANA: High Word - Netlist Identifier Block + 0x002E

Bits	Field Name	Default NVM Value	Description
15:0	Customer IANA - High Word	0x0000	Customer IANA high. Intel to set Intel IANA 0x0000. Customer to fill in their IANA.

3.3.6.1.30 Customer Netlist Version Word - Netlist Identifier Block + 0x0002F

Bits	Field Name	Default NVM 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.

3.3.6.2 Node Header Section

This section describes the structure of the Node Header Section, 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	3.3.6.2.1
0x0001	Node Handle	3.3.6.2.2
0x0002	Node Address	3.3.6.2.3
0x0003	Node Part Number and Node Options	3.3.6.2.4
0x0004	Node I/O Section Pointer	3.3.6.2.5
0x0005	Node Port Options Section Pointer	3.3.6.2.6
0x0006	Node PMD Line Analog Section Pointer	3.3.6.2.7
0x0007	Node PMD Host Analog Section Pointer	3.3.6.2.8

The section is accessible through the GETNODEATTR action starting at the DWord-aligned Attribute ID offset 0x0000. The Node Address within Attribute ID 0x0001 can also be modified through the SETNODEATTR action.

3.3.6.2.1 Node Type and Section Length (0x0000)

Bits	Field Name	Default NVM Value	Description
15:12	Node Type		Defines the node type. Valid values are: 0x0 = PHY device 0x1 = GPIO Controller (Port Expander) 0x2 = MUX Controller (Bus Expander) 0x3 = LED Controller (LED Driver) 0x4 = LED 0x5 = Temperature Sensor 0x6 = Cage (SFP, SFP28, QSFP, QSFP28) 0x7 = Mezzanine Connector 0x8 = ID EEPROM 0x9-0xF = Reserved
11:0	Node Section Length		Node section length in double words. Does not include the first double word, the *Node Section Length* field itself.

3.3.6.2.2 Node Handle (0x0001)

Bits	Field Name	Default NVM 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. Bit 9 = Board Type — Identifies if the node is on the motherboard or a mezzanine card. 0b = LOM 1b = Mezzanine Card For LOM: Bit 8:0 = Unique node identifier within the context of the motherboard. For Mezzanine Cards: Bit 8:6 = Mezzanine card number. Populated by firmware during the mezzanine discovery flow. The value in the mezzanine topology is always 3'b0. Bit 5:0 = Unique node identifier within the context of the mezzanine card.

3.3.6.2.3 Node Address (0x0002)

Bits	Field Name	Default NVM Value	Description
15:14	Reserved		Reserved.
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 = SoC internal bus 01b = I^2C 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 SoC internal bus, the assignment is as follows: 0x000 = PHY Core 0 0x001 = PHY Core 1 0x002 = PHY Core 2 0x003 = I/O Widget 0x004-0x3FF - Reserved For I^2C, the field represents the 7-bit or 10-bit address of the I^2C attached node: 0x000-0x07F = For 7-bit addressable I^2C devices, the 3 high MSB should be set to 0x0. 0x000-0x3FF = For 10-bit addressable I^2C devices. For MDIO, the field represents the 5-bit Port Address: 0x000-0x01F = 5bit MDIO Port Address. 0x020-0x3FF = Reserved.

3.3.6.2.4 Node Part Number and Node Options (0x0003)

Bits	Field Name	Default NVM 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: 0x00 = I/O Widget (Node type - GPIO controller) 0x10 = CEI Connector (Node type - mezzanine connector) 0x11 = SFP+ Cage (Node type - Cage) 0x12 = SFP28 Cage (Node type - Cage) 0x13 = QSFP+ Cage (Node type - Cage) 0x14 = QSFP28 Cage (Node type - Cage) 0x15 = Monochrome LED (Node type - LED) 0x16 = RGB LED (Node type - LED) 0x20 = PCA9545A \ PCA9546A (Node type - MUX controller) 0x21 = PCA9575 (Node type - GPIO controller) 0x22 = PCA9685 (Node type - LED controller) 0x23 = PCA9552 (Node type - LED controller) 0x30 = PHY Core (Node type - PHY) 0x31 = C827 (Node type - PHY) 0x32 = X557-AT4 (Node type - PHY) 0x33 = 88E1543 (Node type - PHY) 0x34 = 88E1512 (Node type - PHY) 0x35 = 88E1514 (Node type - PHY) 0x36 = E810 internal PHY (Node type - PHY) 0x37 = X557-AT2 (Node type - PHY) 0x43 = AT24C32 (ID EEPROM 32K bit) (Node type - ID EEPROM) 0x44 = SFPx Module Temperature Sensor 0x45 = QSFPx Module Temperature Sensor All other values are reserved.
7:4	Reserved		Reserved.
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	Auto Conflict Resolution		Auto conflict resolution. depending on the node type, enables automatic Topology or Media Conflict resolution. Applicable to Innermost PHY and Cage nodes only: • When set on an Innermost PHY node, it enables automatic Topology Conflict resolution. • When set on a Cage node, it enables automatic Media Conflict resolution. Valid for PHY and Cage nodes only. Reserved for all other node types.
0	FW Download Enable		Firmware download enable. 0b = The PHY load its firmware from a dedicated EEPROM/Flash after reset. 1b = The PHY firmware should be downloaded from NVM at initialization Valid for PHY nodes only. Reserved for all other node types.

3.3.6.2.5 Node I/O Section Pointer (0x0004)

Bits	Field Name	Default NVM 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.

3.3.6.2.6 Node Port Options Section Pointer (0x0005)

Bits	Field Name	Default NVM 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.

3.3.6.2.7 Node PMD Line Analog Section Pointer (0x0006)

Bits	Field Name	Default NVM 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.

3.3.6.2.8 Node PMD Host Analog Section Pointer (0x0007)

Bits	Field Name	Default NVM 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.

3.3.6.3 Node I/O Section

This section describes the structure of the Node I/O Section, detailing the I/O connections of a topology node.

Word Offset	Description	Section Reference
0x0000 + n*2	Driving Node Handle	3.3.6.3.1
0x0001 + n*2	I/O Type and Driving Interface	3.3.6.3.2

The section is accessible through the GETNODEATTR action starting at the DWord-aligned Attribute ID offset 0x2000.

3.3.6.3.1 Driving Node Handle (0x0000 + n*2)

Bits	Field Name	Default NVM Value	Description
15:10	Driving I/O Number		The number of the interface with the specified *I/O Type* on the device pointed to by Driving Node Handle 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.

3.3.6.3.2 I/O Type and Driving Interface (0x0001 + n*2)

Bits	Field Name	Default NVM Value	Description
15	Driven	1b	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 *Value* field. 1b = The I/O is driven by the device identified by the Driving Node Handle. Only applicable to I/Os with *I/O Type* of GPIO and MDIO Address. ***Note:*** For I/Os that use inverted I/O function (e.g. RESET_N or INT_N) and the I/O is defined as "Driven", it is driven to the inverse of the *Value* field.
14	Value		For driven I/Os, defines the default value. For strapped I/Os, defines the static value of the I/O is configured to. Only applicable to I/Os with *I/O Type* of GPIO and MDIO Address.
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/O Type* 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/O Type* of GPIO.

Bits	Field Name	Default NVM Value	Description
10:8	I^2C/MDIO Speed		The bus speed. For I^2C, 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/O Type* of I^2C Bus and MDIO Bus.
7:5	I/O Type		The type of the I/O. The defined I/O types are: 000b = PMD 001b = I^2C Bus 010b = MDIO Bus 011b = GPIO 100b = MDIO Address All other values are reserved.
4:0	I/O Function / I/O Number		I/O Function: For I/Os with *I/O Type* 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 0x08 = RX_LOSS 0x09 = RS0 0x0A = RS1 0x0B = EEPROM_WP 0x0C = LED/LED.Red 0x0D = LED.Green 0x0E = LED.Blue 0x0F-0x1F = Reserved I/O Number: For I/Os with *I/O Type* of PMD, I^2C Bus, MDIO Bus, and MDIO Address, 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.

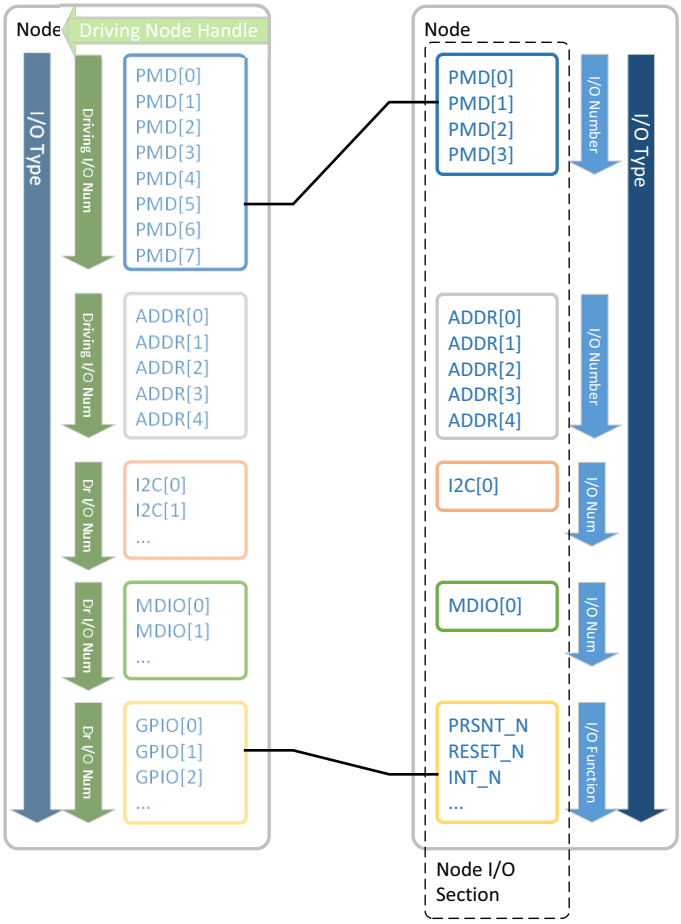


Figure 3-21. I/O Numbering Convention

3.3.6.4 Node Port Option Pointer Section

This section describes the structure of the Node Port Option Pointer Section, detailing the available Port Options.

Word Offset	Description	Section Reference
0x0000 + n	Port Option Pointer	3.3.6.4.1

The section is not accessible through the GETNODEATTR action.

3.3.6.4.1 Node Port Option Pointer (0x0000 + n)

Bits	Field Name	Default NVM Value	Description
15:12	Reserved		Reserved.
11:0	Port Option Pointer		The Port Option's word offset relative to the beginning of the node section.

3.3.6.5 Node Port Option Header Section

This section describes the structure of the Node Port Option Header Section, detailing the available Port Options.

Word Offset	Description	Section Reference
0x0000	Adaptive NVM Global (per Port Option) Super Configuration ID	3.3.6.5.1
0x0001	Adaptive NVM PHY Configuration ID	3.3.6.5.2
0x0002	Minimum SKU	3.3.6.5.3
0x0003	Adaptive NVM PF-to-Port mapping Configuration ID	3.3.6.5.4
0x0004	PMD Count and PHY Capabilities [0] Pointer	3.3.6.5.5
0x0005 + n - 1	PHY Capabilities [n] Pointer	3.3.6.5.6

The section is not accessible through the GETNODEATTR action.

3.3.6.5.1 Adaptive NVM Global (per Port Option) Super Configuration ID (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Adaptive NVM Global (per port option) Super Configuration ID		Global adaptive NVM Super Configuration ID [15:0]. Used to identify the adaptive NVM configuration needed to apply to configure the controller for this Port Option (E810 port mode - 2/4/8 ports; DCB Configuration - pipe monitors, RPB thresholds, VSI/VEB allocation, and so on). See Section 3.3.3.3.

3.3.6.5.2 Adaptive NVM PHY (per Innermost PHY) Super Configuration ID (0x0001)

Bits	Field Name	Default NVM Value	Description
15:0	Adaptive NVMPHY Configuration ID (per Innermost PHY)		Adaptive NVM PHY Configuration ID[15:0]. Used to identify the adaptive NVM configuration ID needed to apply to configure the PHY for the innermost PHY at this Port Option). See Section 3.3.3.3.

3.3.6.5.3 Minimum SKU (0x0002)

Bits	Field Name	Default NVM Value	Description
15:12	Adaptive NVM RDMA & MTU Global Super Configuration ID		Adaptive NVM RDMA & MTU Global Super Configuration ID - Used to identify the adaptive NVM configuration ID for super feature ID (0xF0FD). This super feature controls the RDMA setting together with the MTU setting. The value here must match the number of ports defined in the port option.
11:8	Reserved		Reserved.
7	Required Lane Speed		Maximum lane speed required. 0b = 50G serial required. 1b = 25G serial required.
6:4	Reserved		Reserved.

Bits	Field Name	Default NVM Value	Description
3:2	Required Bandwidth		Specificities the Required bandwidth for this port option: 00b = 200 Gb/s required. 01b = 100 Gb/s required. 10b = 50 Gb/s required. 11b = 25 Gb/s required.
1:0	Required Ports		Number of ports required: 00b = 8 ports required. 01b = 4 ports required. 10b = 2 ports required. 11b = 1 port required.

3.3.6.5.4 Adaptive NVM PF-to-Port Mapping Configuration ID (0x0003)

Bits	Field Name	Default NVM Value	Description
15:0	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.

3.3.6.5.5 PMD Count and PHY Capabilities Pointer (0x0004)

Bits	Field Name	Default NVM 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.

3.3.6.5.6 Capabilities Pointer (0x0005 + n - 1)

Bits	Field Name	Default NVM 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.

3.3.6.6 Node PHY Capabilities Section

This section describes the structure of the Node PHY Capabilities Section, detailing the PHY capabilities for a PHY for a given Port Option. It is used within the PHY node, as described in Section 3.3.6.13.

Word Offset	Description	Section Reference
0x0000	Per-Port Adaptive NVM Configuration ID	3.3.6.6.1
0x0001	Per-Function Adaptive NVM Configuration ID	3.3.6.6.2
0x0002	Per-Port/Function Adaptive NVM Feature ID	3.3.6.6.3
0x0003	PMD Width	3.3.6.6.4
0x0004	Link Options 0	3.3.6.6.5
0x0005	Link Options 1	3.3.6.6.6
0x0006	EEE Options 0	3.3.6.6.7
0x0007	EEE Options 1	3.3.6.6.8
0x0008	Host Side PMD Capabilities 0	3.3.6.6.9
0x0009	Host Side PMD Capabilities 1	3.3.6.6.10
0x000A	Host Side PMD Capabilities 2	3.3.6.6.11
0x000B	Host Side PMD Capabilities 3	3.3.6.6.12
0x000C	Line Side PHY Capabilities (PHY Types) 0	3.3.6.6.13
0x000D	Line Side PHY Capabilities (PHY Types) 1	3.3.6.6.14
0x000E	Line Side PHY Capabilities (PHY Types) 2	3.3.6.6.15
0x000F	Line Side PHY Capabilities (PHY Types) 3	3.3.6.6.16
0x0010	Line Side PHY Capabilities (PHY Types) 4	3.3.6.6.17
0x0011	Line Side PHY Capabilities (PHY Types) 5	3.3.6.6.18
0x0012	Line Side PHY Capabilities (PHY Types) 6	3.3.6.6.19
0x0013	Line Side PHY Capabilities (PHY Types) 7	3.3.6.6.20

The section is accessible through the GETNODEATTR 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.

3.3.6.6.1 Per-Port Adaptive NVM Configuration ID (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Per Port Adaptive NVM 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.

3.3.6.6.2 Per-Function Adaptive NVM Configuration ID (0x0001)

Bits	Field Name	Default NVM Value	Description
15:0	Per Function Adaptive NVM Configuration ID		Per function adaptive NVM Configuration ID [15:0]. Used to identify the adaptive NVM feature needed to apply to configure the port associated with the PMD (PF assignment). The field applies to the Innermost PHY node only and is reserved for all other nodes.

3.3.6.6.3 Per-Port/Function Adaptive NVM Feature ID (0x0002)

Bits	Field Name	Default NVM 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.

3.3.6.6.4 PMD Width (0x0003)

Bits	Field Name	Default NVM 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.

3.3.6.6.5 Link Options 0 (0x0004)

Bits	Field Name	Default NVM 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.
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.

Bits	Field Name	Default NVM Value	Description
10:8	Reserved		Reserved.
7	CL74_ConFEC_Abl		Controls KR FEC (Fire Code FEC) ability advertisement for 25G KR1/CR1. 0b = KR FEC is not supported. 1b = KR FEC is supported.
6	CL91_ConFEC_Abl		Controls RS FEC ability advertisement for 25G KR1/CR1. 0b = RS FEC is not supported. 1b = RS FEC is supported.
5	Reserved		Reserved.
4	RS FEC 544 Request		Controls RS FEC 544 capability request for 25G/50G/100G KR/KR-S/KR1/CR/CR-S/CR1. 0b = RS FEC is not requested. 1b = RS FEC is requested.
3	25G KR FEC Request		Controls KR FEC (Fire Code FEC) capability request for 25G KR/KR-S/KR1/CR/CR-S/CR1. 0b = KR FEC is not requested. 1b = KR FEC is requested.
2	RS FEC 528 Request		Controls RS FEC 528 capability request for 25G KR/KR-S/KR1/CR/CR-S/CR1. 0b = RS FEC is not requested. 1b = RS FEC is requested.
1	10G KR FEC Request		Controls FEC (Fire Code FEC) capability request for 10G KR. 0b = FEC is not requested. 1b = FEC is requested.
0	10G KR FEC Enable		Controls FEC (Fire Code FEC) capability advertisement for 10G KR. 0b = FEC is disabled. 1b = FEC is enabled and is advertised.

3.3.6.6.6 Link Options 1 (0x0005)

Bits	Field Name	Default NVM 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. 0x0 = 10M 0x1 = 100M 0x2 = 1G 0x3 = Reserved 0x4 = 5G 0x5 = 10G 0x6-0xF = Reserved ***Note:*** This option is valid when LPLU is enabled (See bit 3 - LPLU enable).
11	Disable automatic Link on Start-Up		This bit, when set, disables the automatic link on start-up. When the port is defined as manageability port, the LESM ignores this bit and tries to start the link anyway.
10	Reserved		Reserved.
9	Flow Control ASM_DIR Capability		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		Controls Link Flow control PAUSE (A5) ability to advertise. 0b = PAUSE is advertised as 0. 1b = PAUSE is advertised as 1.

Bits	Field Name	Default NVM Value	Description
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). This option is valid, when the LESM is enabled. 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 module. 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.
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.

3.3.6.6.7 EEE Options 0 (0x0006)

Bits	Field Name	Default NVM Value	Description
15:0	EEE_TWsysTX		EEE_TWsysTX. See Section 5.3.1.

3.3.6.6.8 EEE Options 1 (0x0007)

Bits	Field Name	Default NVM Value	Description
15:11	Reserved		Reserved.
10	EEE Enable 100GBASE-KR2-PAM4	0b	Controls the EEE functionality for 100GBASE-KR2-PAM4. 0b = EEE is disabled. 1b = EEE is enabled.
9	EEE Enable 100GBASE-KR4	0b	Controls the EEE functionality for 100GBASE-KR4. 0b = EEE is disabled. 1b = EEE is enabled.
8	EEE Enable 50GBASE-KR-PAM4	0b	Controls the EEE functionality for 50GBASE-KR-PAM4. 0b = EEE is disabled. 1b = EEE is enabled.

Bits	Field Name	Default NVM Value	Description
7	EEE Enable 50GBASE-KR2	0b	Controls the EEE functionality for 50GBASE-KR2. 0b = EEE is disabled. 1b = EEE is enabled.
6	Reserved		Reserved.
5	EEE Enable 25GBASE-KR	0b	Controls the EEE functionality for 25GBASE-KR. 0b = EEE is disabled. 1b = EEE is enabled.
4	EEE Enable 10GBASE-KR	0b	Controls the EEE functionality for 10GBASE-KR. 0b = EEE is disabled. 1b = EEE is enabled.
3	EEE Enable 1000BASE-KX	0b	Controls the EEE functionality for 1000BASE-KX. 0b = EEE is disabled. 1b = EEE is enabled.
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	0b	Controls the EEE functionality for 100BASE-TX. 0b = EEE is disabled. 1b = EEE is enabled.

3.3.6.6.9 Host Side PMD Capabilities 0 (0x0008)

Host-side PMD capabilities are mapped based on line side link speed.

Bits	Field Name	Default NVM Value	Description
15	50GBASE-KR2		
14:13	Reserved		Reserved.
12	25G-AUI-C2C		
11	25GBASE-KR1		
10	25GBASE-KR-S		
9	25GBASE-KR		
8	10G-SFI-C2C		
7	10GBASE-KR/CR1		
6	10G-SFI-DA		
5	5GBASE-KR		
4:3	Reserved		Reserved.
2	1G-SGMII		
1	1000BASE-KX		
0	100M-SGMII		

3.3.6.6.10 Host Side PMD Capabilities 1 (0x0009)

Host-side PMD capabilities are mapped based on line side link speed.

Bits	Field Name	Default NVM Value	Description
15:14	Reserved		Reserved.
13	200G-AUI8		
12	200G-AUI4		
11	200GBASE-KR4-PAM4		
10	100G-AUI2		
9	Reserved		Reserved.
8	100GBASE-KR2-PAM4		
7	100GBASE-KP4-PAM4		
6	100G-AUI4		
5	100G-CAUI4		
4	100GBASE-KR4		
3	50G-AUI1		
2	50GBASE-KR-PAM4		
1	50G-AUI2		
0	50G-LAUI2		

3.3.6.6.11 Host Side PMD Capabilities 2 (0x000A)

Host-side PMD capabilities are mapped based on line side link speed.

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Reserved.

3.3.6.6.12 Host Side PMD Capabilities 3 (0x000B)

Host-side PMD capabilities are mapped based on line side link speed.

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Reserved.

3.3.6.6.13 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 3.3.6.6.9.

Bits	Field Name	Default NVM Value	Description
15	10GBASE-LR		
14	10GBASE-SR		
13	10G-SFI-DA		
12	10GBASE-T		
11	5GBASE-KR		
10	5GBASE-T		
9:7	Reserved		Reserved.
6	1G-SGMII		
5	1000BASE-KX		
4	1000BASE-LX		
3	1000BASE-SX		
2	1000BASE-T		
1	100M-SGMII		
0	100BASE-TX		

3.3.6.6.14 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 3.3.6.6.10.

Bits	Field Name	Default NVM Value	Description
15:14	Reserved		Reserved.
13	25G-AUI-C2C		
12	25G-AUI-AOC/ACC		
11	25GBASE-KR1		
10	25GBASE-KR-S		
9	25GBASE-KR		
8	25GBASE-LR		
7	25GBASE-SR		
6	25GBASE-CR1		
5	25GBASE-CR-S		
4	25GBASE-CR		
3	25GBASE-T		
2	10G-SFI-C2C		
1	10G-SFI-AOC/ACC		
0	10GBASE-KR/CR1		

3.3.6.6.15 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 3.3.6.6.11.

Bits	Field Name	Default NVM Value	Description
15	50GBASE-LR		
14	50GBASE-FR		
13	50GBASE-SR		
12	50GBASE-CP		
11	50G-AUI2		
10	50G-AUI2-AOC/ACC		
9	50G-LAUI2		
8	50G-LAUI2-AOC/ACC		
7	50GBASE-KR2		
6:5	Reserved		
4	50GBASE-CR2		
3:0	Reserved		Reserved.

3.3.6.6.16 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 3.3.6.6.12.

Bits	Field Name	Default NVM Value	Description
15	100GBASE-DR		
14	100GBASE-SR2		
13	100GBASE-CR2-PAM4		
12	100GBASE-KP4-PAM4		
11	Reserved		Reserved.
10	100G-AUI4		
9	100G-AUI4-AOC/ACC		
8	100G-CAUI4		
7	100G-CAUI4-AOC/ACC		
6	100GBASE-KR4		
5	100GBASE-LR4		
4	100GBASE-SR4		
3	100GBASE-CR4		
2	50G-AUI1		
1	50G-AUI1-AOC/ACC		
0	50GBASE-KR-PAM4		

3.3.6.6.17 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	Default NVM Value	Description
15	400GBASE-FR8	0b	
14	200G-AUI8	0b	
13	200G-AUI8-AOC/ACC	0b	
12	200G-AUI4	0b	
11	200G-AUI4-AOC/ACC	0b	
10	200GBASE-KR4-PAM4	0b	
9	200GBASE-DR4	0b	
8	200GBASE-LR4	0b	
7	200GBASE-FR4	0b	
6	200GBASE-SR4	0b	
5	200GBASE-CR4-PAM4	0b	
4	100G-AUI2		
3	100G-AUI2-AOC/ACC		
2:1	Reserved		Reserved.
0	100GBASE-KR2-PAM4		

3.3.6.6.18 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	Default NVM Value	Description
15:4	Reserved		Reserved.
3	400G-AUI8	0b	
2	400G-AUI8-AOC/ACC	0b	
1	400GBASE-DR4	0b	
0	400GBASE-LR8	0b	

3.3.6.6.19 Line Side PHY Capabilities (PHY Types) 6 (0x0012)

The following table defines the Line Side PHY Capabilities for the outermost PHY. Not defined for PMDs with the *Outermost PHY* bit not set.

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	Reserved.

3.3.6.6.20 Line Side PHY Capabilities (PHY Types) 7 (0x0013)

The following table defines the Line Side PHY Capabilities for the outermost PHY. Not defined for PMDs with the *Outermost PHY* bit not set.

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	Reserved.

3.3.6.7 Node PMD Analog Section

This section describes the structure of the Node PMD Analog Section, detailing the analog configuration of a PHY Lane.

Word Offset	Description	Section Reference
0x0000 + n*14	1G/5G C2C Coefficients Low	3.3.6.7.1
0x0001 + n*14	Polarity and 1G/5G C2C Coefficients High	3.3.6.7.2
0x0002 + n*14	10G C2C Coefficients Low	3.3.6.7.3
0x0003 + n*14	10G C2C Coefficients High	3.3.6.7.4
0x0004 + n*14	10G C2M Coefficients Low	3.3.6.7.5
0x0005 + n*14	10G C2M Coefficients High	3.3.6.7.6
0x0006 + n*14	25G C2C Coefficients Low	3.3.6.7.7
0x0007 + n*14	25G C2C Coefficients High	3.3.6.7.8
0x0008 + n*14	25G C2M Coefficients Low	3.3.6.7.9
0x0009 + n*14	25G C2M Coefficients High	3.3.6.7.10
0x000A + n*14	50G C2C Coefficients Low	3.3.6.7.11
0x000B + n*14	50G C2C Coefficients High	3.3.6.7.12
0x000C + n*14	50G C2M Coefficients Low	3.3.6.7.13
0x000D + n*14	50G C2M Coefficients High	3.3.6.7.14

Notes: 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 which they are specifying, as well as the PHY-specific script which configures the device.

C2C Coefficients are used for link modes directly connecting PHY devices, such as 10G SFI DA and 25G AUI C2C. C2M coefficients are used for link modes that use a module, such as 10GBASE-LR and 25G-AUI AOC/ACC

This section is accessible through the GETNODEATTR action starting at the DWord-aligned Attribute ID offset 0x6800 for the line side and 0x7000 for the host side.

3.3.6.7.1 1G/5G C2C Coefficients Low (0x0000 + n*14)

Defines the transmit equalizer settings for 1G/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 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	Default NVM Value	Description
15:8	C[0][7:0]		Cursor value for C2C 1G/5G speeds.
7:0	C[1][7:0]		Post-Cursor value for C2C 1G/5G speeds.

3.3.6.7.2 Polarity and 1G/5G C2C Coefficients High (0x0001 + n*14)

Defines the lane polarity and transmit equalizer settings for 1G/5G serial AUI/C2C interfaces.

Bits	Field Name	Default NVM 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	TXFFE Override 50G AUI-C2M		TXFFE Override 50G AUI-C2M control. 0b = Use silicon default TXFFE values for AUI-C2M PHY modes with 50G lane speeds. 1b = Use netlist defined TXFFE values for AUI-C2M PHY modes with 50G lane speeds.
12	TXFFE Override 50G AUI-C2C		TXFFE Override 50G C2C control. 0b = Use silicon default TXFFE values for AUI-C2C PHY modes with 50G lane speeds. 1b = Use netlist defined TXFFE values for AUI-C2C PHY modes with 50G lane speeds.
11:10	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 TXFFE Override bit corresponding to each mode.
8	TXFFE Override 1G/5G		TXFFE Override 1G/5G control. 0b = Use silicon default TXFFE values for PHY modes with 1G/5G lane speeds. 1b = Use netlist defined TXFFE values for PHY modes with 1G/5G lane speeds.
7:0	C[-1][7:0]		Pre-Cursor value for C2C 1G/5G speeds.

3.3.6.7.3 10G C2C Coefficients Low (0x0002 + n*14)

Defines the transmit equalizer settings for 10G serial C2C interfaces.

Bits	Field Name	Default NVM 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.

3.3.6.7.4 10G C2C Coefficients High (0x0003 + n*14)

Defines the transmit equalizer settings for 10G serial AUI-C2C interfaces.

Bits	Field Name	Default NVM Value	Description
15:9	Reserved		Reserved.
8	TXFFE Override 10G AUI-C2C		TXFEE Override 10G AUI-C2C control. 0b = Use silicon default TXFFE values for AUI-C2C PHY modes with 10G lane speeds. 1b = Use netlist defined TXFFE values for AUI-C2C PHY modes with 10G lane speeds.
7:0	C[-1][7:0]		Pre-Cursor value for 10G speeds.

3.3.6.7.5 10G C2M Coefficients Low (0x0004 + n*14)

Defines the transmit equalizer settings for 10G serial AUI-C2M interfaces.

Bits	Field Name	Default NVM 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.

3.3.6.7.6 10G C2M Coefficients High (0x0005 + n*14)

Defines the transmit equalizer settings for 10G serial AUI-C2M interfaces.

Bits	Field Name	Default NVM Value	Description
15:9	Reserved		Reserved.
8	TXFFE Override 10G AUI-C2M		TXFFE Override 10G AUI-C2M control. 0b = Use silicon default TXFFE values for AUI-C2M PHY modes with 10G lane speeds. 1b = Use netlist defined TXFFE values for AUI-C2M PHY modes with 10G lane speeds.
7:0	C[-1][7:0]		Pre-Cursor value for C2M 10G speeds.

3.3.6.7.7 25G C2C Coefficients Low (0x0006 + n*14)

Defines the transmit equalizer settings for 25G serial AUI-C2C interfaces.

Bits	Field Name	Default NVM Value	Description
15:8	C[0][7:0]		Cursor value for AUI-C2C 25G speeds.
7:0	C[1][7:0]		Post-Cursor value for AUI-C2C 25G speeds.

3.3.6.7.8 25G C2C Coefficients High (0x0007 + n*14)

Defines the transmit equalizer settings for 25G serial AUI-C2C interfaces.

Bits	Field Name	Default NVM Value	Description
15:9	Reserved		Reserved.
8	TXFFE Override 25G AUI-C2C		TXFFE Override 25G AUI-C2C control. 0b = Use silicon default TXFFE values for AUI-C2C PHY modes with 25G lane speeds. 1b = Use netlist defined TXFFE values for AUI-C2C PHY modes with 25G lane speeds.
7:0	C[-1][7:0]		Pre-Cursor value for AUI-C2C 25G speeds.

3.3.6.7.9 25G C2M Coefficients Low (0x0008 + n*14)

Defines the transmit equalizer settings for 25G serial AUI-C2M interfaces.

Bits	Field Name	Default NVM Value	Description
15:8	C[0][7:0]		Cursor value for AUI-C2M 25G speeds.
7:0	C[1][7:0]		Post-Cursor value for AUI-C2M 25G speeds.

3.3.6.7.10 25G C2M Coefficients High (0x0009 + n*14)

Defines the transmit equalizer settings for 25G serial AUI-C2M interfaces.

Bits	Field Name	Default NVM Value	Description
15:9	Reserved		Reserved.
8	TXFFE Override 25G AUI-C2M		TXFFE Override 25G AUI-C2M control. 0b = Use silicon default TXFFE values for AUI-C2M PHY modes with 25G lane speeds. 1b = Use netlist defined TXFFE values for AUI-C2M PHY modes with 25G lane speeds.
7:0	C[-1][7:0]		Pre-Cursor value for AUI-C2M 25G speeds.

3.3.6.7.11 50G C2C Coefficients Low (0x000A + n*14)

Defines the transmit equalizer settings for 50G serial AUI-C2C interfaces.

Bits	Field Name	Default NVM Value	Description
15:8	C[0][7:0]		Cursor value for AUI-C2C 50G speeds.
7:0	C[1][7:0]		Post-Cursor value for AUI-C2C 50G speeds.

3.3.6.7.12 50G C2C Coefficients High (0x000B + n*14)

Defines the transmit equalizer settings for 50G serial AUI-C2C interfaces.

Bits	Field Name	Default NVM Value	Description
15:8	C[-2][7:0]		Pre-Cursor 2 (C[-2]) value for AUI-C2C 50G speeds.
7:0	C[-1][7:0]		Pre-Cursor 1 (C[-1]) value for AUI-C2C 50G speeds.

3.3.6.7.13 50G C2M Coefficients Low (0x000C + n*14)

Defines the transmit equalizer settings for 50G serial AUI-C2M interfaces.

Bits	Field Name	Default NVM Value	Description
15:8	C[0][7:0]		Cursor value for AUI-C2M 50G speeds.
7:0	C[1][7:0]		Post-Cursor value for AUI-C2M 50G speeds.

3.3.6.7.14 50G C2M Coefficients High (0x000D + n*14)

Defines the transmit equalizer settings for 50G serial AUI-C2M interfaces.

Bits	Field Name	Default NVM Value	Description
15:8	C[-2][7:0]		Pre-Cursor 2 (C[-2]) value for AUI-C2M 50G speeds
7:0	C[-1][7:0]		Pre-Cursor 1 (C[-1]) value for AUI-C2M 50G speeds

3.3.6.8 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 and line sides.

Word Offset	Description	Section Reference
0x0000	PMD Analog Misc Length	3.3.6.8.1
0x0001	Auto-neg LESM Timeout	3.3.6.8.2
0x0002	1G/5G/10G serial AUI LESM Timeout	3.3.6.8.3
0x0003	25G/50G serial AUI LESM Timeout	3.3.6.8.4

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.

3.3.6.8.1 PMD Analog Misc Length (0x0000)

Bits	Field Name	Default NVM Value	Description
15:8	Reserved		Reserved.
7:0	PMD Analog Misc Length		The length of the PMD analog miscellaneous section in words.

3.3.6.8.2 Auto-Neg LESM Timeout (0x0001)

Bits	Field Name	Default NVM Value	Description
15:8	Auto-Neg LESM Timeout		Auto-negotiation LESM timeout defined in 200 ms increments.
7:0	Reserved		Reserved.

3.3.6.8.3 1G/5G/10G Serial AUI LESM Timeout (0x0002)

Bits	Field Name	Default NVM Value	Description
15:8	10G Serial AUI LESM Timeout		10G AUI LESM timeout defined in 200 ms increments.
7:0	1G/5G Serial AUI LESM Timeout		1G/5G AUI LESM timeout defined in 200 ms increments.

3.3.6.8.4 25G/50G Serial AUI LESM Timeout (0x0003)

Note: This parameter applies to all link modes that are using 50G serial signaling. For example, 100G implemented as 2x50G.

Bits	Field Name	Default NVM Value	Description
15:8	50G Serial AUI LESM Timeout		50G AUI LESM Timeout defined in 200 ms increments.
7:0	25G Serial AUI LESM Timeout		25G AUI LESM Timeout defined in 200 ms increments.

3.3.6.9 Node LED Configuration Section

This section describes the structure of the Node LED Configuration 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. See Section 3.3.6.9.3.

Word Offset	Description	Section Reference
0x0000	Port Option Port Affinity Low	3.3.6.9.1
0x0001	Port Option Port Affinity High	3.3.6.9.2
0x0002 + n*1	Port Option Color and Condition [n], where n = 0:3	3.3.6.9.3

The section is accessible through the GETNODEATTR action starting at the DWord-aligned Attribute ID offset 0xA000.

3.3.6.9.1 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 action, the *Port Affinity* bits always refer to Physical MAC Ports.

Bits	Field Name	Default NVM Value	Description
15	Port 15		If set, the LED is associated with Port 15
14	Port 14		If set, the LED is associated with Port 14
13	Port 13		If set, the LED is associated with Port 13
12	Port 12		If set, the LED is associated with Port 12
11	Port 11		If set, the LED is associated with Port 11
10	Port 10		If set, the LED is associated with Port 10
9	Port 9		If set, the LED is associated with Port 9
8	Port 8		If set, the LED is associated with Port 8
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

3.3.6.9.2 Port Option Port Affinity High (0x0001)

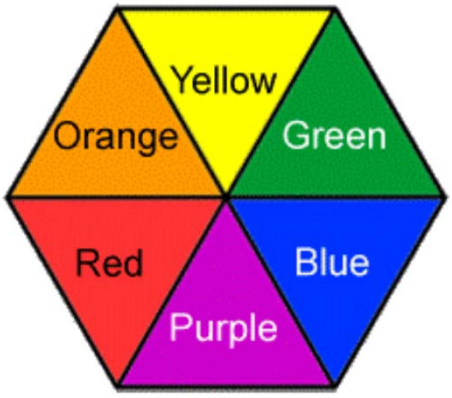
Bits	Field Name	Default NVM Value	Description
15:14	Logic Operator	10b	The logic operator to be used between port flags. 00b = Reserved 10b = AND 10b = OR 11b = Reserved
13:9	Reserved		Reserved.
8	Port ID		If set, the LED is used for port identification based on software request.
7:4	Reserved		Reserved.
3	Port 19		If set, the LED is associated with Port 19.
2	Port 18		If set, the LED is associated with Port 18.
1	Port 17		If set, the LED is associated with Port 17.
0	Port 16		If set, the LED is associated with Port 16.

3.3.6.9.3 Port Option Color and Condition [n] (0x0004 + 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	Default NVM Value	Description
15:14	Reserved		Reserved.
13:11	Blink		Blink configuration for the LED. 000b = No Blink 001b = Blink 010b = Blink on MAC Activity All other values are reserved. Blinking results in a cadence of 200 ms on / 200 ms off.

Bits	Field Name	Default NVM Value	Description
10:8	Color		Defines the color of an RGB LED. The color is mixed though driving the associated Red, Green and Blue LEDs with the appropriate PWM signals through a PCA9585 LED controller. The possible colors are as follows: 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  Colors are only supported if the Node Part Number is RGB LED. The node has three I/Os defined, and they all are driven by the same PCA9585 LED controller.
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	Default NVM 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 0x07 = LINK_LESS_THAN_1G 0x08 = Reserved 0x09 = LINK_LESS_THAN_5G 0x0A = LINK_LESS_THAN_10G 0x0B = LINK_LESS_THAN_25G 0x0C = Reserved 0x0D = LINK_LESS_THAN_50G 0x0E = LINK_LESS_THAN_100G 0x0F = LINK_LESS_THAN_200G 0x10 = LINK_LESS_THAN_400G 0x11 = LINK_100M 0x12 = LINK_1G 0x13 = Reserved 0x14 = LINK_5G 0x15 = LINK_10G 0x16 = LINK_25G 0x17 = Reserved 0x18 = LINK_50G 0x19 = LINK_100G 0x1A = LINK_200G 0x1B = LINK_400G 0x1C = NO_LINK 0x1E:0x1F = Reserved 0x20 = TEMP_WARN_N 0x21 = TEMP_CRIT_N 0x22 = FAN_ON_AUX 0x23:0x3F - Reserved

3.3.6.10 Node Thermal Configuration Section

This section describes the structure of the Node Thermal Configuration Section, detailing the configuration of the temperature sensor.

Word Offset	Description	Section Reference
0x0000	Min and Max Readable Value	3.3.6.10.1
0x0001	Tolerance and Hysteresis	3.3.6.10.2
0x0002	Resolution Low	3.3.6.10.3
0x0003	Resolution High	3.3.6.10.4
0x0004	Offset Low	3.3.6.10.5
0x0005	Offset High	3.3.6.10.6
0x0006	Normal Max and Warning High Thresholds	3.3.6.10.7
0x0007	Critical High and Fatal High Threshold	3.3.6.10.8
0x0008 + n*2	Temperature Sensor Port Affinity Low	3.3.6.10.9
0x0009 + n*2	Temperature Sensor Port Affinity High	3.3.6.10.10

The section is accessible through the GETNODEATTR 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.

Note: For SFP and QSFP standard sensors, E810 firmware follows the standard and should not modify parameters from the netlist. The firmware should not take (i.e. ignore) the critical thresholds from the netlist, where SFP or QSFP standard sensor is defined. Instead, it should use the thresholds as read from the module itself (SFF-8436 defines them in bytes 128-129)

3.3.6.10.1 Min and Max Readable Value (0x0000)

Bits	Field Name	Default NVM Value	Description
15:8	Max Readable Value	0xFF	Thermal Sensor max readable value.
7:0	Min Readable Value	0x00	Thermal Sensor max readable value.

3.3.6.10.2 Tolerance and Hysteresis (0x0001)

Bits	Field Name	Default NVM Value	Description
15:8	Hysteresis	0x02	Thermal Sensor Hysteresis (°C).
7:0	Tolerance	0x01	Thermal Sensor *plusTolerance* and *minusTolerance* (°C).

3.3.6.10.3 Resolution Low (0x0002)

Bits	Field Name	Default NVM 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. ***Note:*** For the internal thermal sensor definition, the resolution is not taken from the netlist, but it is set to the NVM default.

3.3.6.10.4 Resolution High (0x0003)

Bits	Field Name	Default NVM 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. ***Note:*** For the internal thermal sensor definition, the resolution is not taken from the netlist, but it is set to the NVM default.

3.3.6.10.5 Offset Low (0x0004)

Bits	Field Name	Default NVM Value	Description
15:0	Offset [15:0]	0x0000	Lower two bytes of the Thermal Sensor offset. The field uses a real32 representation.

3.3.6.10.6 Offset High (0x0005)

Bits	Field Name	Default NVM Value	Description
15:0	Offset [31:16]	0x0000	Higher two bytes of the Thermal Sensor offset. The field uses a real32 representation.

3.3.6.10.7 Normal Max and Warning High Thresholds (0x0006)

Bits	Field Name	Default NVM Value	Description
15:8	Warning High		Warning High - High temperature warning threshold (°C).
7:0	Normal Max		Normal Max - Maximum normal temperature threshold (°C).

3.3.6.10.8 Critical High and Fatal High Thresholds (0x0007)

Bits	Field Name	Default NVM Value	Description
15:8	Fatal High		Fatal High - Fatal temperature alarm threshold (°C). ***Note:*** For any thermal sensor, this threshold must be set between the "Internal Thermal Sensor Minimum Threshold" and "Internal Thermal Sensor Maximum Threshold" NVM parameters.
7:0	Critical High		Critical High - Critical temperature alarm threshold (°C).

3.3.6.10.9 Temperature Sensor Port Affinity Low (0x0008 + 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 will always refer to Physical MAC Ports.

Bits	Field Name	Default NVM Value	Description
15	Port 15		If set, the temperature sensor is associated with Port 15
14	Port 14		If set, the temperature sensor is associated with Port 14
13	Port 13		If set, the temperature sensor is associated with Port 13
12	Port 12		If set, the temperature sensor is associated with Port 12
11	Port 11		If set, the temperature sensor is associated with Port 11
10	Port 10		If set, the temperature sensor is associated with Port 10
9	Port 9		If set, the temperature sensor is associated with Port 9
8	Port 8		If set, the temperature sensor is associated with Port 8
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

3.3.6.10.10 Temperature Sensor Port Affinity High (0x0009 + n*2)

Bits	Field Name	Default NVM Value	Description
15:8	Sensor Location		Defines the location of the sensor.
7:4	Reserved		Reserved.
3	Port 19		If set, the temperature sensor is associated with Port 19.
2	Port 18		If set, the temperature sensor is associated with Port 18.
1	Port 17		If set, the temperature sensor is associated with Port 17.
0	Port 16		If set, the temperature sensor is associated with Port 16.

3.3.6.11 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	3.3.6.11.1
0x0001	Reserved	3.3.6.11.2

3.3.6.11.1 Parent Node Handle (0x0000)

Bits	Field Name	Default NVM 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 Parent Node Handle field should be ignored. 1b = The node is associated with a parent node. The parent node is identified by the Parent Node Handle.
9:0	Parent Node Handle		The node handle of the parent node.

3.3.6.11.2 Reserved (0x0001)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Reserved.

3.3.6.12 Node Scratch Section

This section describes the structure of the Node Scratch Section. Used by scripts to store node-related temporary variables. This section is implemented only in EMP firmware RAM and is not stored in the NVM. The exact use of the scratch section is subject to script implementation and it is beyond the scope of this specification. The section must be initialized to 0x0000 on POR, EMPR, and script engine reset.

Word Offset	Description	Section Reference
0x0000	Scratch 0	3.3.6.12.1
0x0001	Scratch 1	3.3.6.12.2
0x0002	Scratch 2	3.3.6.12.3
0x0003	Scratch 3	3.3.6.12.4
0x0004	Scratch 4	3.3.6.12.5
0x0005	Scratch 5	3.3.6.12.6

The section is accessible through the GETNODEATTR and SETNODEATTR action starting at the DWord-aligned Attribute ID offset 0x3800.

3.3.6.12.1 Scratch 0 (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Reserved.

3.3.6.12.2 Scratch 1 (0x0001)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Reserved.

3.3.6.12.3 Scratch 2 (0x0002)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Reserved.

3.3.6.12.4 Scratch 3 (0x0003)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Reserved.

3.3.6.12.5 Scratch 4 (0x0004)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Reserved.

3.3.6.12.6 Scratch 5 (0x0005)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Reserved.

3.3.6.13 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 and line side PMD Analog Sections, host 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 Adaptive NVM Pointer of the active Port Option on the innermost PHY node identifies the Adaptive NVM Feature required to configure the controller 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.

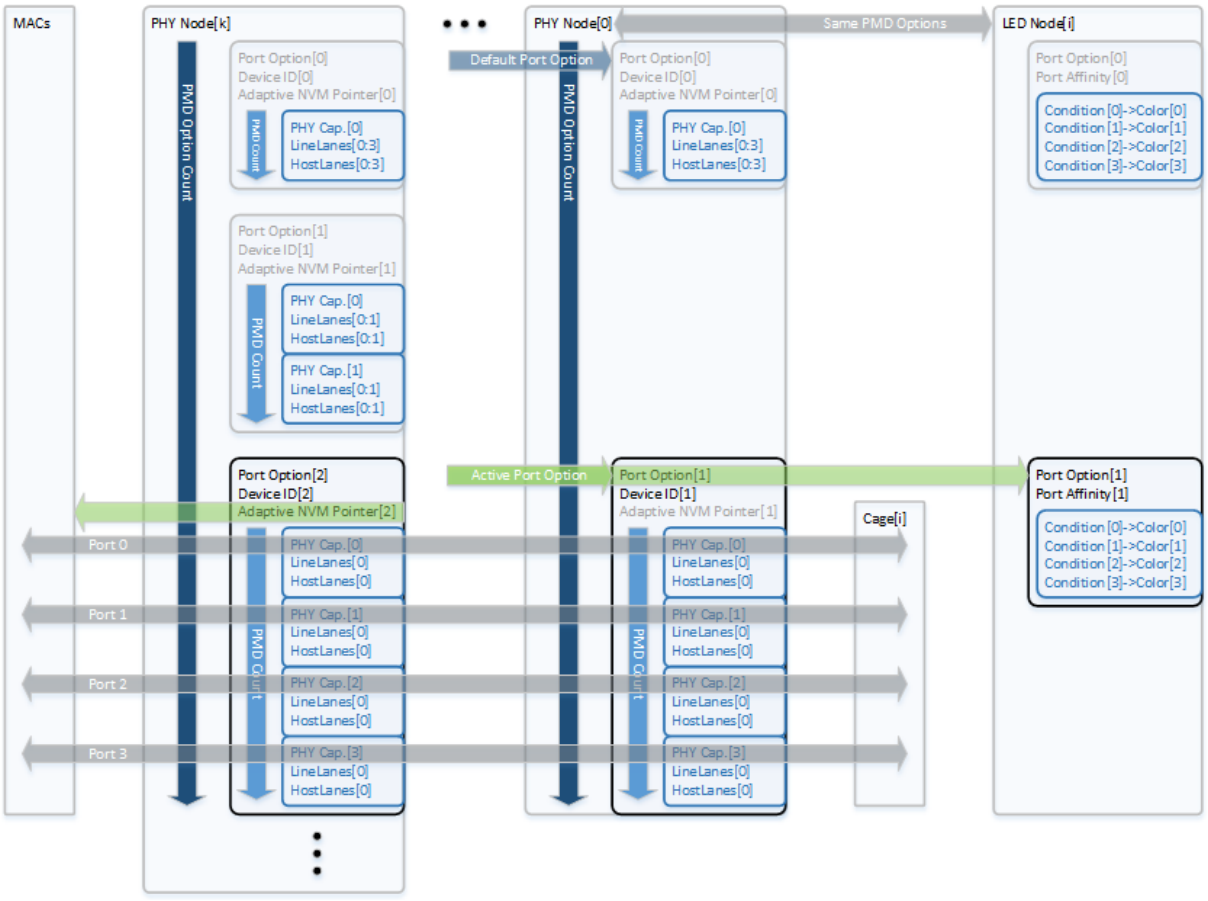


Figure 3-22. Port Options

Word Offset	Attribute ID	Description	Section Reference
0x0000	0x0000	Node Header Section	3.3.6.2
0x0008	0x2000	Node I/O Section	3.3.6.3
	0x6800	PMD Line Analog Section	3.3.6.7
	0x8800	PMD Line Analog Misc Section	3.3.6.8
	0x7000	PMD Host Analog Section	3.3.6.7
	0x9000	PMD Host Analog Misc Section	3.3.6.8
	N/A	Port Option Pointer Section	3.3.6.4
	N/A	Port Option Header Section [0:Port Option Count]	3.3.6.5
	0x4000	PHY Capabilities Section [0:PMD Count - 1][0:Port Option Count]	3.3.6.6

The Node I/O Section should define the following I/Os: I^2C 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.

3.3.6.14 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	3.3.6.2
0x0008	0x2000	Node I/O Section	3.3.6.3
N/A	0x3800	Node Scratch Section	3.3.6.12

The Node I/O Section should define the following I/Os: I^2C, Int_N, and Reset_N.

3.3.6.15 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	3.3.6.2
0x0008	0x2000	Node I/O Section	3.3.6.3
N/A	0x3800	Node Scratch Section	3.3.6.12

The Node I/O Section should define the following I/Os: I^2C and Reset_N.

3.3.6.16 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	3.3.6.2
0x0008	0x2000	Node I/O Section	3.3.6.3
N/A	0x3800	Node Scratch Section	3.3.6.12

The Node I/O Section should define the following I/Os: I^2C and Reset_N.

3.3.6.17 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 3.3.6.2). 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 3.3.6.2) 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	3.3.6.2
0x0008	0x2000	Node I/O Section	3.3.6.3
0x000A / 0x000E	N/A	Node Parent Section	3.3.6.11
0x000C + n*6 / 0x00010 + n*6	0xA000	Node LED Configuration Section. n - port option index	3.3.6.9
N/A	0x3800	Node Scratch Section	3.3.6.12

3.3.6.18 Temperature Sensor Node

This section describes the block structure of an 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	3.3.6.2
0x0008	0x2000	Node I/O Section	3.3.6.3
0x000C	N/A	Node Parent Section	3.3.6.11
0x0010	0xC000	Node Thermal Configuration Section	3.3.6.10

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 3.3.6.2) 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: I^2C, Int_N, and Reset_N.

3.3.6.19 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	3.3.6.2
0x0008	0x2000	Node I/O Section	3.3.6.3

The Node I/O Section should define the following I/Os: I^2C and EEPROM_WP.

3.3.6.20 Cage Node

This section describes the block structure of an 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	3.3.6.2
0x0008	0x2000	Node I/O Section	3.3.6.3
N/A	0x3800	Node Scratch Section	3.3.6.12

The Node Header Section should indicate a *Bus Type* of I^2C and the *Bus Address* should be 0xA0.

For an SFP+ or SFP28 cage, the following I/Os must be defined: I^2C, Present_N. Optionally, the following I/Os could be also defined: TX_Disable.

For a QSFP+ or QSFP28 cage, the following I/Os must be defined: I^2C, Present_N, Reset_N, Int_N. Optionally, the following I/Os could be also defined: Modsel_N, LPMode.

3.3.6.21 Mezzanine Connector Node

This section describes the block structure of an Mezzanine Connector 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	3.3.6.2
0x0008	0x2000	Node I/O Section	3.3.6.3

The Node Header Section should indicate a *Bus Type* of I^2C and the *Bus Address* should be 0xA8.

Setting the path to this mezzanine connector and using this bus allows for reading the mezzanine card's ID EEPROM.

For a CEI mezzanine connector, part of the motherboard topology, the following I/Os must be defined: I^2C, Present_N, Int_N, Reset_N, MDIO, ADDR[0:2], and PMD[0:3].

When the topology of a mezzanine card is discovered, the connections with the motherboard are determined by matching the I/O types of the mezzanine connector from the mezzanine card topology with the I/Os of the mezzanine connector of the motherboard topology according to the below in Figure 3-23.

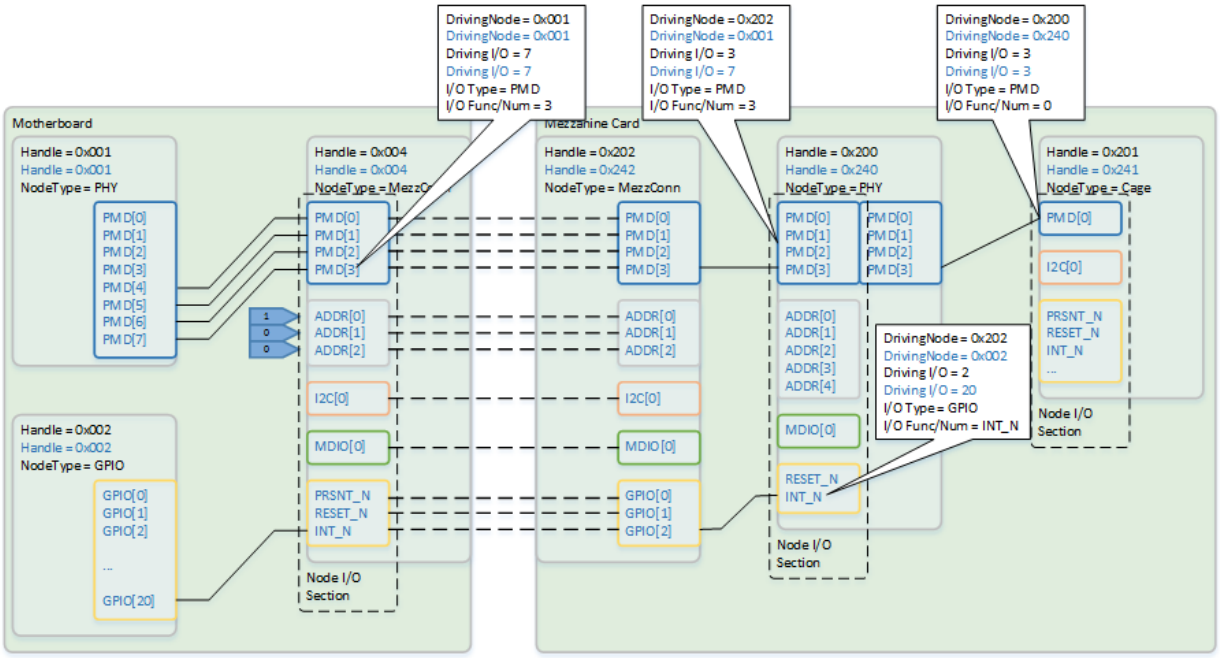


Figure 3-23. Mezzanine Connector Mapping

3.3.7 Topology Netlist Constraints and Conventions

The topology netlist describes design-specific configuration and connections between various motherboard or mezzanine card components, referred to as nodes.

The E810 link management engine imposes the following constraints:

3.3.7.1 Netlist Size Constraints

- The maximum size of the netlist supported in EMP firmware 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 NVM is 28 KB.
- The maximum size of the mezzanine card netlist in the IDEEPROM is 4 KB.
- The maximum size of the motherboard netlist in RAM is 12 KB. For all PHY and LED nodes, this includes only the currently-active Port Option.
- 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.

3.3.7.2 Node Constraints

- The maximum number of nodes supported on a motherboard is 128.
- The maximum number of nodes supported on a mezzanine card is 32.
- The maximum number of nodes of Mezzanine Connector type supported on a motherboard is 5.
- The maximum number of nodes of Mezzanine Connector type supported on a mezzanine card is 1.
- The maximum depth of cascaded PHY nodes (PHY chain length) is 4.

3.3.7.3 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 mezzanine card is 64.
- 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 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 motherboard is 20.
- 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 motherboard is 16.
- The maximum number of I/O connections with I/O Type of INT_N supported on a mezzanine card is 4.
- 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 the 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.
- With the exception of I/O connections with *I/O Type* of I^2C Bus, MDIO Bus and GPIO with *I/O Function* of RESET_N must be dedicated point to point connections. Another exception to this rule is GPIO signal driving the rate select pins of modules (RS0 and RS1).
- If a node is accessible through an I^2C or MDIO, the connection must be described by the first entry in the I/O connections.
- On each board the I^2C attached MUX Controllers, GPIO Controllers, and LED Drivers must have a shared reset signal. Each of these nodes must have and 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.

- Mezzanine cards that implements more than one PHY, should not share their reset signal.
- The maximum number of MDIO/I^2C I/Os on the I/O Widget node is 16.
- The maximum number of MDIO/I^2C I/Os on a Mux node is 62.

3.3.7.4 Node I/O Conventions

3.3.7.4.1 I/O Widget

- Pins MDC[0:4]/SCL[0:5], MDIO[0:4]/SDA[0:4] are referred to as MDIO[0:4] or I^2C[0:4], depending on the *I/O Type* of the connected I/O connection.
- Pins GPIO[0:31] are referred to as GPIO[0:31].

3.3.7.4.2 PCA9545A

- Pins SD0 and SC0 defined in Chapter 6 of the PCA9545A Datasheet are referred to as I^2C[0].
- Pins SD1 and SC1 defined in Chapter 6 of the PCA9545A Datasheet are referred to as I^2C[1].
- Pins SD2 and SC2 defined in Chapter 6 of the PCA9545A Datasheet are referred to as I^2C[2].
- Pins SD3 and SC3 defined in Chapter 6 of the PCA9545A Datasheet are referred to as I^2C[3].

3.3.7.4.3 PCA9575

- Pins P0_0 through P0_7 defined in Table 3 of the PCA9575 Datasheet are referred to as GPIO[0:7].
- Pins P1_0 through P1_7 defined in Table 3 of the PCA9575 Datasheet are referred to as GPIO[8:15].

3.3.7.4.4 PCA9685

- Pins LED0 through LED15 defined in Table 3 of the PCA9685 Datasheet are referred to as GPIO[0:15].

3.3.7.4.5 C827

- Pins LIP0, LIN0, LOP0, and LON0 defined in Table 2 of the C827 Datasheet are referred to as PMD[0].
- Pins LIP1, LIN1, LOP1, and LON1 defined in Table 2 of the C827 Datasheet are referred to as PMD[1].
- Pins LIP2, LIN2, LOP2, and LON2 defined in Table 2 of the C827 Datasheet are referred to as PMD[2].
- Pins LIP3, LIN3, LOP3, and LON3 defined in Table 2 of the C827 Datasheet are referred to as PMD[3].
- Pins HIP0, HIN0, HOP0, and HON0 defined in Table 2 of the C827 Datasheet are referred to as PMD[0].

- Pins HIP1, HIN1, HOP1, and HON1 defined in Table 2 of the C827 Datasheet are referred to as PMD[1].
- Pins HIP2, HIN2, HOP2, and HON2 defined in Table 2 of the C827 Datasheet are referred to as PMD[2].
- Pins HIP3, HIN3, HOP3, and HON3 defined in Table 2 of the C827 Datasheet are referred to as PMD[3].

3.3.7.5 PHY Node Constraints

- The maximum number of lanes on line or host side of a PHY node is 8.
- The maximum number of Port Options supported for a PHY node is 16.
- The E810 has one PHY node defined for each of the three PHY Cores.
- The first two PHY Core nodes allow for a total of eight line-side lanes, while the third one only four line-side lanes. None of the PHY Core nodes have host-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.

3.3.7.6 GPIO Controller Node Constraints

- • The E810 has one GPIO Controller node defined for the I/O Widget.
- The netlist must capture the GPIO nodes in a well-defined order. The first node captured by the netlist must be the I/O Widget with a node handle equal to 0.

3.3.7.7 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.

3.3.7.8 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.

3.3.7.9 Cage Node Constraints

- The I/O connection with *I/O Type* of I^2C must point to an I^2C interface dedicated to the cage. All cages must have dedicated I^2C interfaces. The MODSEL_N pin of the QSFP cages must be asserted low through a strapping.

3.3.8 Link Topology Admin Commands

Most components of the link topology are accessed by the device firmware, either directly or using PHY configuration scripts. However, sometimes there is a need for direct access of the software driver, or tools driver to the link topology components. As link topology components are owned by the firmware, any such access should be coordinated and conducted thorough firmware admin command, where the firmware can synchronize and control the required access.

The following link topology firmware admin commands are defined:

Table 3-53. Link topology Admin Commands

Command	Opcode	Description	Section Reference
Set Port Identification LED	0x06E9	Set the LED used for identification of this port.	3.3.8.1

3.3.8.1 Set Port Identification LED (0x06E9)

The Set Port Identification LED admin command is used to set the LED that is used to identify the port as indicated in the topology structures (see Section 3.3.6.9.2). The software driver should provide the Logical Port Number.

Table 3-54. Set Port Identification LED Admin Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 for details.
Opcode	2-3	0x06E9	Command opcode.
Datalen	4-5	0x0	No external buffer for this command.
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 firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Logical Port Number	16-17		Software uses this field to send the logical port number. Software might mark the logical port number as not valid. In this case, firmware might use the port that is owned by the function. **Byte 16**: Logical Port number This field specifies the port number, and it is used when the physical function owns more than one port or when the physical function owns single port, but the *Logical Port number is valid* bit (Bit 17.0) is turned on. **Bit 17.0:** Logical Port number is valid **All other bits** = Reserved.
Set Indent Mode	18		**Bit 18.0:** 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. **All other bits** = Reserved.
Reserved	19-31		Reserved. Set to zero.

3.4 Non-Volatile Memory (NVM)

3.4.1 General Overview

The following conditions specific to the E810 induced an approach to NVM access:

- LAN and RDMA 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 E810's various virtualization capabilities (like SR-IOV with VFs and PFs) can virtually reach into 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 E810 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.

3.4.1.1 Requirements on NVM Access

The basic requirements from NVM access in the E810 include the following:

- Guarantee that only Intel-provided firmware code (EMP, PE, PHY) is run and device-critical data is used by the E810.
  - The firmware code and device-critical data is cryptographically signed and authenticated prior to execution.
  - The Intel ECSS/EDSS is used for signing.
  - PKCS #1 v2.2 format with 2048-bit RSA keys and SHA-256 hash is used for signing EMSA-PKCS1-v1.5.
- Protect NVM image against compromised supply chains and at least partial protection against physical tampering. Full protection against physical tampering 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. That is, ROM-Miniload-NVM image.
  - Authentication is only performed at POR/EMPR resets and it is not required on every NVM access.
  - The hash value of the public key is stored in fuses, while the public key is part of the NVM image.
- Compromised key revocation is supported, while only level two keys can be revoked by providing a new CA certificate. The key for CA authentication can not be changed in the field.

- Protection against rollback to the NVM version with known security issues is provided.
  - This is achieved by guaranteeing that the NVM image security revision is not decreased during NVM update.
  - A way to override the security revision rollback protection with BMC manageability commands is provided.
- 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.
- 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.
- Protect NVM update flow from power failure before completion. This implies the image-update procedure of modules uses a double-bank policy.
- Meet the boot time requirements described in PCIe specification.
- Reduce the surface for security attacks and protect as much NVM content a possible taking the "white list" approach. Consequently, only a small PFA area with system-specific settings surviving NVM update is unprotected by design.
- 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.
- Prevent a malicious software from causing permanent damage to the system, and to the NVM (and the E810 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.

3.4.1.2 Operational Limitations

The NVM protection method selected in the E810 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 CSRs, which are auto-loaded 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, the 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.

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 auto-load process still occurs, and the state of the device might not be the hardware default state.

3.4.2 External Flash

In the E810, the LAN controller core is the main client of the internal SPI controller. The E810'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 an NVM image.

3.4.2.1 E810 NVM Regions

In the E810, the NVM uses a 16 or 32 MB Flash and contains the following regions:

- **Descriptor region** — The first 4 KB of the Flash that contains the following data:
  - — Signature
  - — Content section (3 DWords)
  - — Component section
  - — Region section
  - — Masters 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 can 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.

3.4.3 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:

- Software must be able to partially update the module without being forced to rewrite the entire module (like SMBus address, VPD, and so on).

  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 E810 commits the entire updated Shadow RAM into the Flash.

- On device-level resets (in contrast to function-level resets), the module (or parts of it) is auto-loaded by the E810 into registers that are mapped to the host memory BAR.

  Auto-load 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 milliseconds) 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 E810 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 3-24, 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 E810 is set by hardware default and enters into blank Flash programming mode (refer to Section 3.4.4.2).

Following a Power-on Reset (POR), the E810 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 Figure 3-24 and Figure 3-25), 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 E810 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.

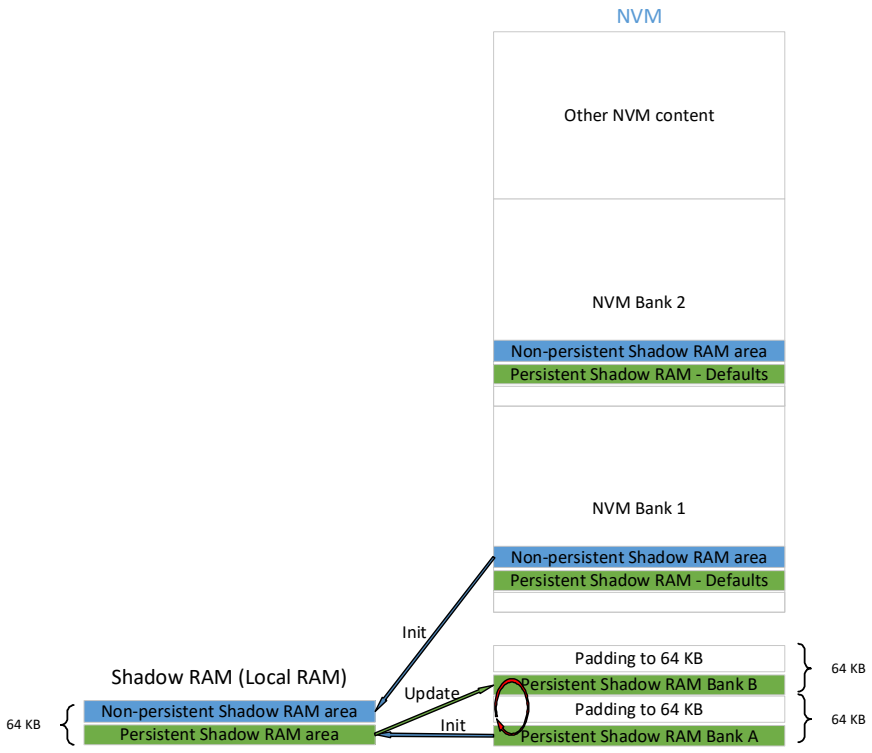


Figure 3-24. NVM Shadow RAM

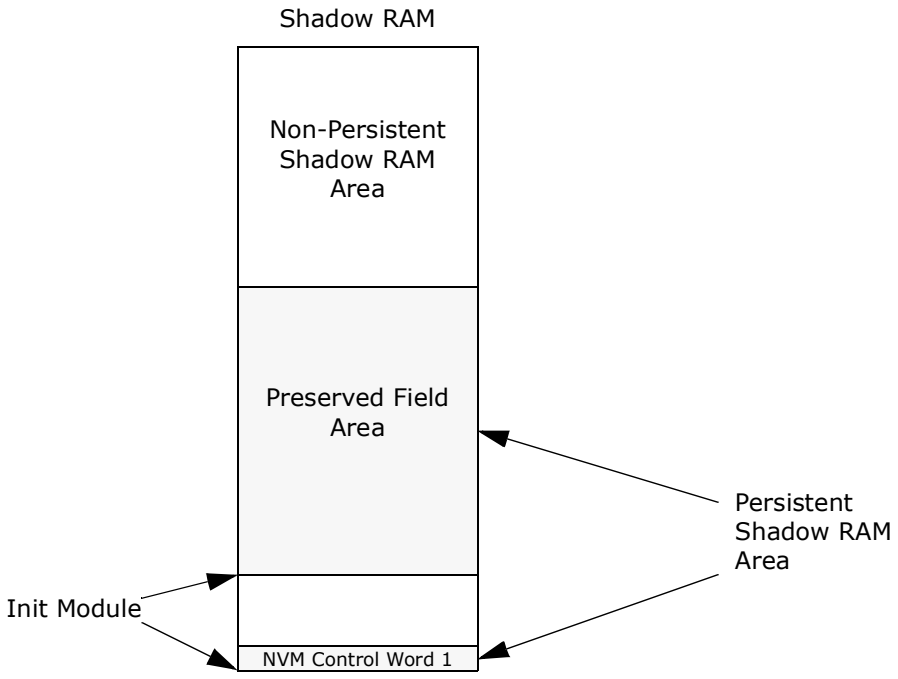


Figure 3-25. Shadow RAM Structure

3.4.4 NVM Access Modes

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

The NVM is accessed in multiple ways according to the entity and the state of the NVM image. following are the different access modes.

- SPI access mode
- Normal access mode
  - — Through admin commands.
  - — VPD register set
  - — Memory map
- Blank Flash programming mode.

3.4.4.1 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 (see Section 9.5.13.6). This prevents the following situations:

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

3.4.4.1.1 Normal Read Access

Memory-mapped read accesses to the NVM do not require the EMP to be 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 GLPCI_LBARCTRL.*FLASH_EXPOSE* CSR bit).

3.4.4.1.2 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 E810, although PCIe transactions are completed normally.

3.4.4.2 Blank Flash Programming Mode

The E810 enters local blank Flash programming mode based on the following:

- When a blank Flash is detected. It means that the NVM *Blank 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 E810.
- 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 safe 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 E810 is in this mode (see conditions in the previous list), the EMP/PE codes are not loaded from the NVM and the EMP/PE remains disabled. The E810 is not able to exchange any kind of traffic over the lines and no admin command can be posted. The E810 is in an unknown operational state where only the Flash programming flow is operational.

3.4.5 NVM Update Flows

The flows described in this section affect only normal programming mode. When in the Blank Flash Programming mode, refer to the NVM access procedures described in Section 3.4.8.

It is firmware's responsibility to keep track of Shadow RAM consistency between Internal RAM and Flash. If GLOBR/CORER occurs while Shadow RAM is inconsistent, EMPR is triggered, which eventually results in coherent Shadow RAMs by rolling-back the internal RAM changes.

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 9-46). Failing to complete an update (by setting *LAST* bit in the command flag) within that time, results in EMPR which causes 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.

3.4.5.1 Flash High-Level Map

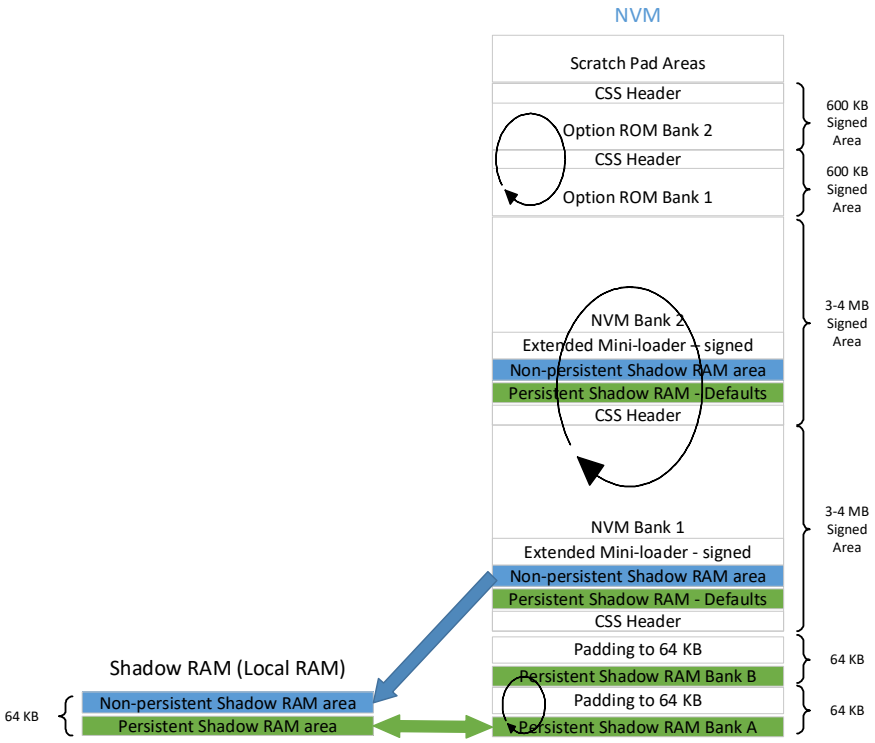


Figure 3-26. Flash High-Level Map

Figure 3-26 shows the high level mapping of the NVM in the E810, 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 might 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 3.4.5.3 and Section 3.4.5.4, and Section 3.4.5.6 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, containing the code to be run by EMP embedded processor.
  - — PE Image area, containing the code to be run by the Protocol Engine.
  - — PHY Configuration scripts.
  - — Parts CORER and GLOBR reset register auto-load sections, which are not modifiable by the Adaptive NVM features.

  - — EMP Global and Setting modules, containing basic configurations common to all images.
  - — Adaptive NVM metadata, which is a data structure containing the metadata required to modify an image to match a given configuration.
  - — PHY Analog modules (internal Ethernet PHY microcode and configuration) loaded by the E810 at power-up events only.
  - — PE Settings module, containing defaults to PE registers.
  - — Recovery Mode firmware image, separately signed. See Section 15.3.

  See Section 3.4.5.6 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 3.4.5.5 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.

Refer to Section 6.3 for the detailed NVM map.

3.4.5.2 Generic Flows

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

3.4.5.2.1 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 9.5.13.7.
   b. If the check was successful (identical), the EMP validates the new bank and invalidates the old bank.
      1. 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.
      2. 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.
      3. The current (old) bank is invalidated by setting its *Validity* field to 00b.
      4. Firmware clears internal bit indicating Flash and Shadow RAM are inconsistent.

3.4.5.3 VPD Update

3.4.5.3.1 First VPD Area Programming

The VPD capability is exposed on the PCIe interface only if the GLPCI_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.

3.4.5.3.2 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 E810.

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).
   — 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.
   a. The EMP takes ownership over the NVM resources for a write.
   b. EMP performs the dump as described in Section 3.4.5.2.1.
   c. 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 indicate that the flow needs to be restarted from step 1 at this stage by successive VPD write accesses initiated by VPD software.
   d. The EMP releases the NVM ownership.

Notes: Users must be made aware that dumping the VPD change into the Flash might take a few hundredths of a millisecond 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 E810 has just started a Shadow RAM dump (Step 6), then it might be that the write request times out.

3.4.5.4 Updating Items in the Preserved Fields Area (PFA)

Figure 3-27 describes the PFA update flow:

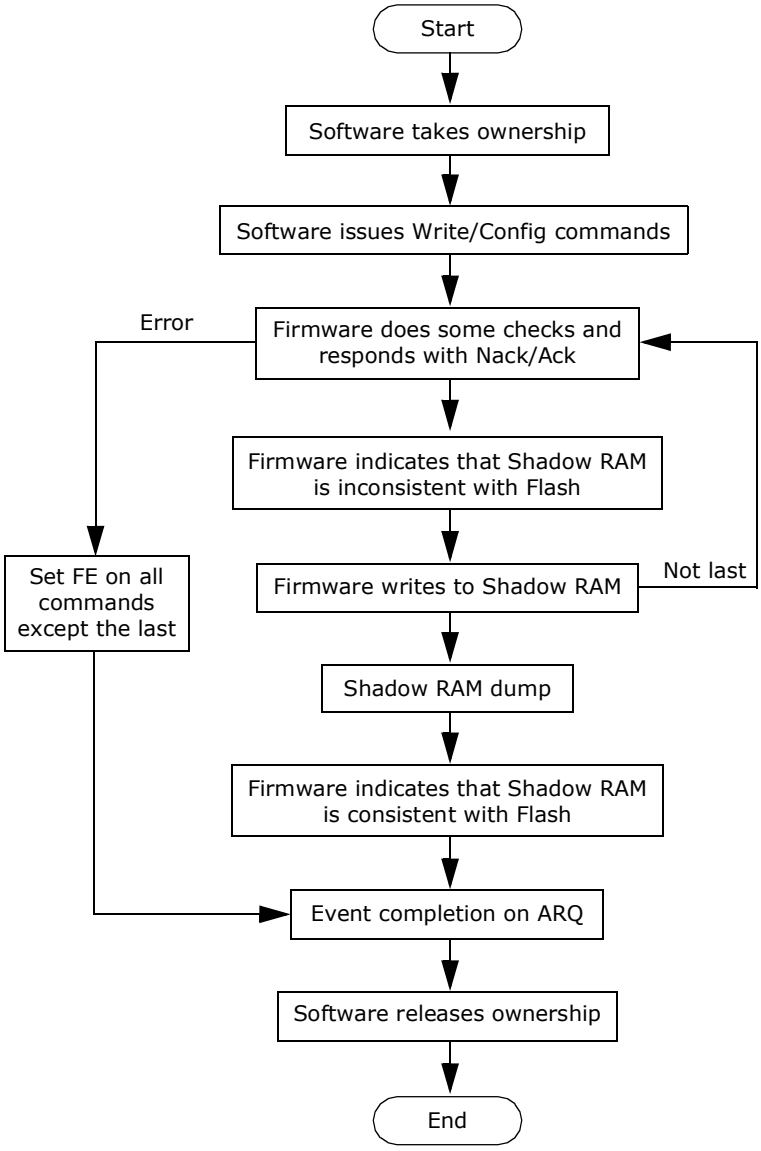


Figure 3-27. PFA Update Flow

The update flow detailed description is as follows:

1. Software takes ownership over the NVM resource for a write (see Section 9.5.13.6).
2. Software issues one or several NVM Write commands or NVM Config Write commands. For NVM Write admin command see Section 3.4.10.3. For NVM Config Write admin command see Section 3.4.10.5.

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, 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.

   - 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 3.4.5.2.1). 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.

   - 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 9.5.13.7).
12. Software initiates a reset for loading the modifications into the E810.

3.4.5.4.1 Limitations on PFA Updates/Handling

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

- When new TLVs are added, they must be added at the end of the PFA, replacing part of the padding TLV.
- Firmware ignores unknown TLVs and does not treat their presence as an error.
- Reserved fields in existing TLVs must not be reused.
- The size of TLVs must not change between versions. One exception is the ANVM feature selection TLV, which may expand into the next padding area.
- TLVs are never removed from PFA, even if no longer used.

3.4.5.5 Scratch Pads Update

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

- Firmware Scratch Pad Area (pointed by NVM word 0x50)
- Link Topology Scratch Pad Area (pointed by NVM word 0x4B)

Figure 3-28 shows the non-authenticated module update flow:

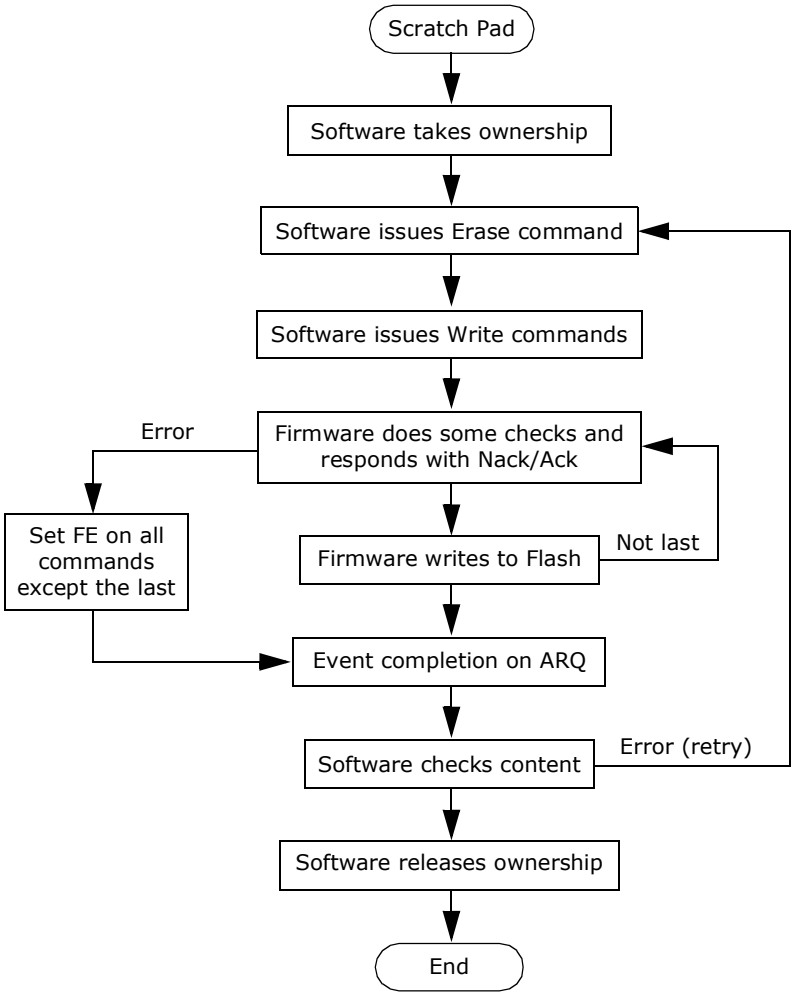


Figure 3-28. Non-Authenticated Module Update Flow

The flow details are below:

1. Software takes ownership over the NVM resource for a write (see Section 9.5.13.6).
2. Software issues a single NVM Erase command to the EMP, providing the following parameters (see Section 3.4.10.2):
   - 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 3.4.10.3):
   - 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.
   - 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.
6. Software releases NVM ownership (see Section 9.5.13.7).
7. Software takes ownership over the NVM resource for a read (see Section 9.5.13.6).
8. Software reads from Flash the contents of the new module to check that it has been correctly recorded into the Flash.
   - In case it was not, Software re-does the flow.
9. Software releases NVM ownership (see Section 9.5.13.7).
10. Software initiates a device reset (PCIR, CORER, or GLOBR) for loading the modifications into the E810.

3.4.5.6 Updating a Double Bank Module

The following flow supports only 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. This module is not authenticated.

Figure 3-29 shows the double banked modules update flow:

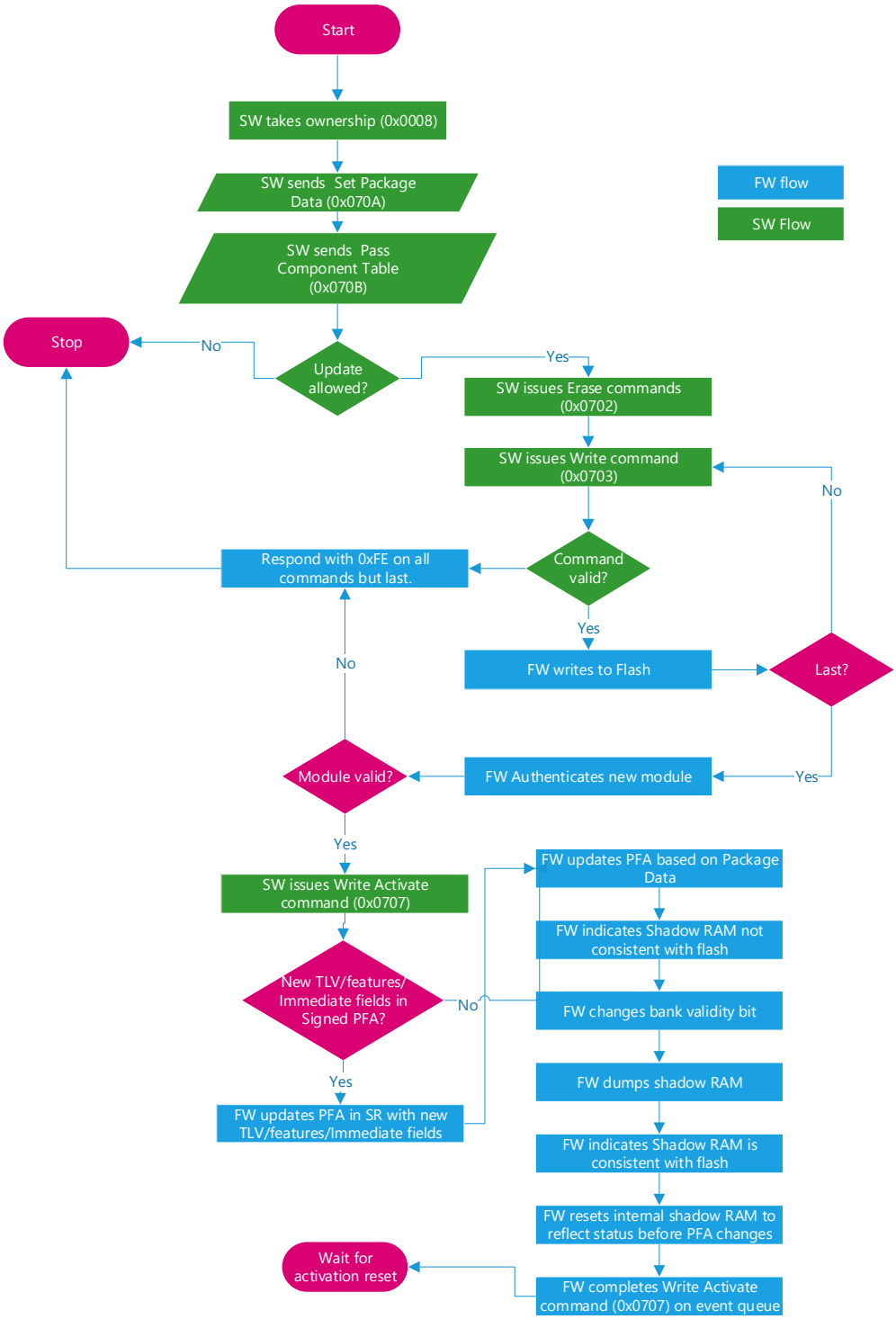


Figure 3-29. Authenticated Module Update Flow

The detailed flow is described as follows:

1. Software Tools verifies that the source/destination NVM images are compatible:
   a. Takes ownership on the NVM (see Section 9.5.13.6) and reads valid NVM sections through admin commands (see Section 3.4.10.1).
   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.

   Note: This step is not performed by EEUpdate, LANConf and NUR-a tools, which are not using config files. These tools are used by OEMs and Hardware manufacturers only (not end-users), who refer to Intel documentation for compatibility check.
2. If not done previously, software takes ownership over the NVM resource for a write (see Section 9.5.13.6).
3. Software issues a single NVM Erase command to the EMP, providing the following parameters (see Section 3.4.10.2):
   - 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 3.4.10.3):
   - 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 4 for the next (elementary) command of the sequence.

7. If the *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 3.4.9. 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 Failed* or the *Module Signature Check Failed* bit). The flow goes to Step 22.
   a. The *lad_srev* in the destination is equal or greater than the one in the source.
   b. The *Blank NVM Device ID* field in the destination corresponds to the E810 device ID (0x1590).
   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 3.4.5.6.1.
   e. These checks are not done for the netlist module.

   Note: Note that before authenticating the Option ROM module, the module must be appended with the last 330 words of the module's area, as the CSS header has been moved to the module's trailer. See Section 6.1.5.2 for more details.
8. Firmware posts an event completion on ARQ to software.
   a. Where applicable, software driver must save *RESET_FLAG* as returned by this command.
9. Software driver can 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 results in rollback of the previous write so that authentication must be done again on new write completion.
10. A Write command that modifies PFA TLV configuration (including VPD) can be called with the *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 the 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 is written by default and does not require setting any flag. Only banks that were written completely (that is, the *Last* bit was set) can be activated. Firmware posts a fail response to the command if one of the following checks fails:
    a. All specified banks were written.
    b. If an NVM bank was written, it is also activated.
13. Firmware sets internal bit indicating that Flash and Shadow RAM are inconsistent.
14. If the command was issued from ACQ: From this point until command completion (Step 21), in case of a CORER/GLOBR/PFR, firmware triggers EMPR.
15. Firmware parses the content of the package data provided by the Set Package Data command and modifies the PFA accordingly.
16. 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).
17. 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 is not 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 runs the ANVM flow to ensure that 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 is rejected.

18. The firmware changes the valid bit indication in Control Word 1.
19. The firmware dumps the Shadow RAM to Flash (see Section 3.4.5.2.1).
20. Firmware clears internal bit indicating that Flash and Shadow RAM are inconsistent.
21. The firmware posts an event completion on ARQ to software. It posts a completion/response to the last admin command of the sequence.
22. Software releases NVM ownership (see Section 9.5.13.7).
23. At this stage, if a new update is needed, the software can 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.
24. Software checks the *RESET_FLAG* field.
25. Software Tools checks the value of the *RESET_FLAG* field in the event posted by firmware:
    - If *RESET_FLAG* == PERST, Software Tools notifies the user to reboot the system since it no ability to issue a PERST cycle (it can only ask the OS to do so).
    - 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 feedback to BMC with the requested reset to activate the update.
26. Software may choose to activate the new Firmware in two ways:
    - 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 can take a longer time to complete and might violate PCIe rules.
    - Send a NVM Update EMPR command, and then request for an immediate PERST event. In this case, the NVM Update EMPR command creates 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.**
27. Auto-load (driven by new Firmware) occurs from the new SR contents.
28. Software Tools checks that the destination map 'EETrackID' was loaded in Internal SR to verify that there was no spurious reset during update, causing ROM to load the wrong Shadow RAM bank.
    a. In case the original map 'EETrackID' is read from Internal SR, **Software Tools** restarts the flow from the beginning.
29. (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.

3.4.5.6.1 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, 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 two GFIDs: an original GFID and a current one. The following algorithm is used to define if an image can be updated based on the two GFIDs in the current (Adapter) and next (Update) images:

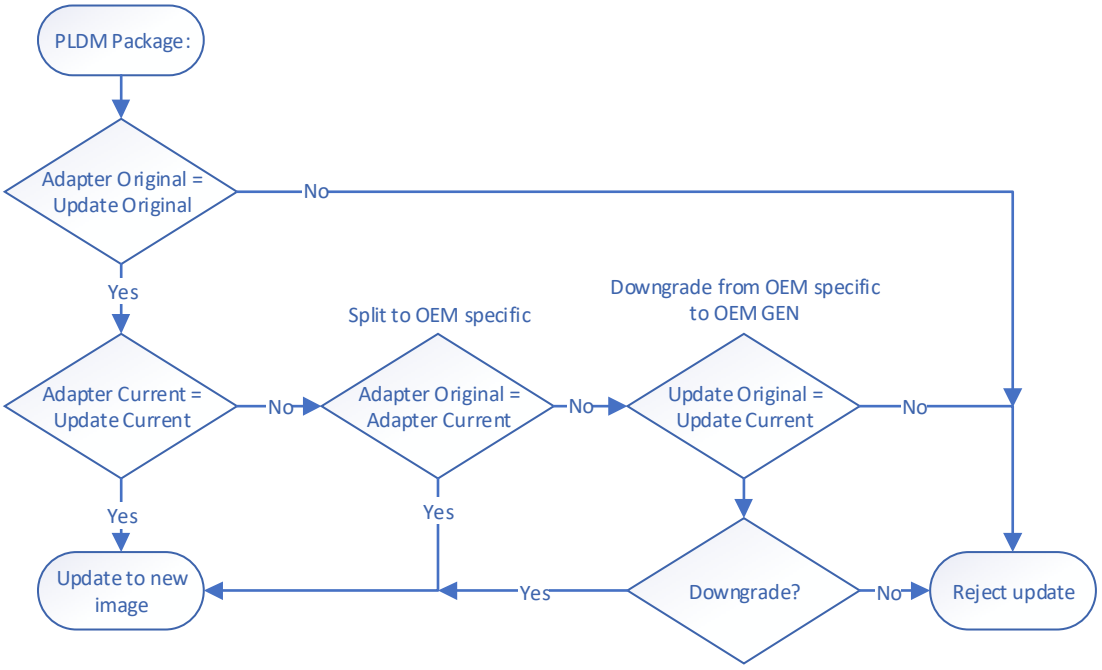


Figure 3-30. GFID Handling Flow

The GFIDs are stored in the non-PFA part of the Shadow RAM for the current image and in the package data of the PLDM header for the new image.

3.4.5.6.2 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.

3.4.5.7 Flash Wear-Out Protection

This 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 (years)*365(days/year)*24(hours)*60(minutes)/100,000 = ~53 minutes.

The E810 does not have access to the real-time clock, and its clocks do not survive a POR reset. As a result, the algorithm has some limitations related to POR resets. Assuming a POR is not a frequent event that 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 Figure 3-31 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 3-31.

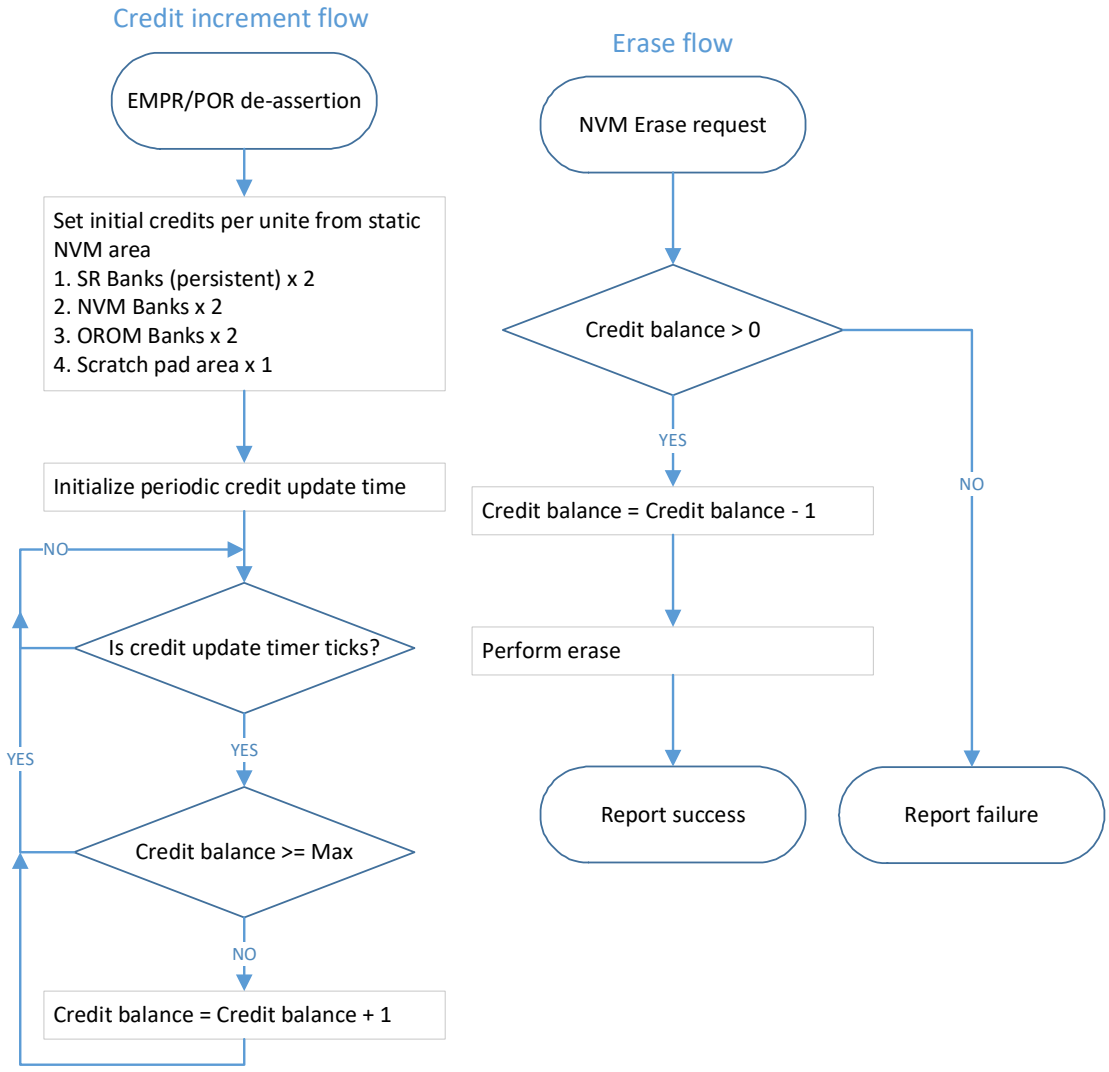


Figure 3-31. Flash Wear-Out Protection Algorithm[1]

1. Credit update time is once per hour.

3.4.5.8 Save Factory Settings AQC

This section describes the flow executed by the Save Factory Settings Command (see Section 3.4.10.9).

3.4.5.8.1 Save Factory Settings Flow

The following flow is run by the Save Factory Settings admin command (opcode 0x0708) and can be used by manufacturing tools to modify the image before programming the device.

- Word pointer to PFA in internal Shadow RAM (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 second 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 bytes of Factory Settings:
   - password = 0x2811
   - Size = 32B + PFA size + netlist size
   - TLV_Extension_Offset = 32B + PFA size
   - TLV_Extension_Size = netlist size
   - CIR_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'

3.4.6 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, and so on), drivers, EMP, PE, BMC (via EMP), and VPD software.

Table 3-55 lists the different accesses to the NVM.

Table 3-55. Clients and Access Types to the NVM

Client	NVM Access Method	Accessed Programmed 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 E810 into the Flash.
PF Software	Parallel via Memory (CSR) BAR (32-bit read)	Flash Part	0x000000 - 0xFFFFFF	The address is relative to the beginning of the Flash. See Section 13.1.2.1 for details of the Flash offset within the CSR BAR.
	Parallel via Option ROM BAR (32-bit read only).	Flash Part	0x020000 - 0xFFFFFF	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 3.4.10.

3.4.6.1 Memory-Mapped Host Interface

The Flash is read by the E810 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 E810 is in blank Flash programming mode. The Flash size exposed is 16 MB. Refer to Section 13.1.1.2 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 1 MB by default. The E810 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, and 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 E810 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, the Option ROM BAR is disabled.

The E810 controls accesses to the Flash when it decodes a valid access. Out of range memory-mapped read access returns arbitrary data.

Refer to Section 3.1.2.2.1 for details on memory/Option ROM BAR access rules.

Notes:

- Flash read accesses are assembled by the E810 each time the access is greater than a byte-wide access.
- The E810 byte reads to the Flash take about 2 to 30 µs. The E810 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 9.5.13.6.

3.4.7 Flash Access Contention

Flash read accesses initiated through PFs might occur concurrently to the firmware modifying the NVM contents. The E810 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 take/release flows for any read or write access to NVM (see Section 9.5.13.6 and to Section 9.5.13.7 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 acquire/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 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 the responsibility of hardware/firmware to update the NVM according to the Flash update sequence described in Section 3.4.5.3.
- 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 takes 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 rebooting.
- The BMC delays PERST# de-assertion or boot running until after the BMC completes 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 ensure that another user is not currently initiating a local host reboot there.

- The firmware's responsibility 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.

3.4.8 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 9.5.13.6 and Section 9.5.13.7 for the NVM ownership taking/releasing procedures as well for the associated timeouts.

3.4.8.1 Auto-Load

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

Firmware initialization performed at power-up and EMPR events consists for three steps, while DIGEST updating (discussed in Section 3.4.8.1.2) 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. Auto-load time-critical NVM section, which is 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 auto-load on every PCIe, CORER, and GLOBR resets.
   b. Initialize all firmware code being ready to handling all runtime events.
   c. Report initialization completion to the host software.

Simplified NVM Bank and Extended mini-Loader maps are shown in Figure 3-32 for better understanding of the initialization flows.

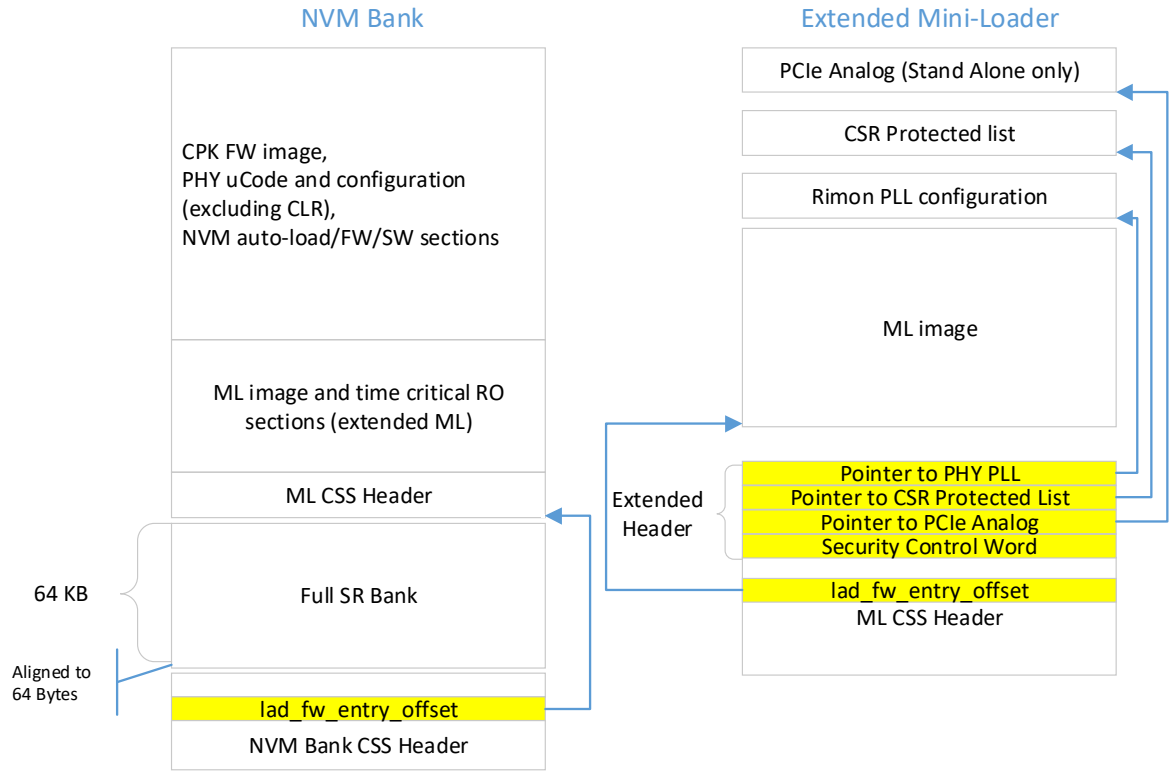


Figure 3-32. Simplified NVM Bank and Extended Mini-Loader Maps

3.4.8.1.1 EMP Firmware Init Flow

The firmware is ready to PERST de-assertion handing 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.

3.4.8.1.2 Firmware Measurement for Remote Attestation

A new *Controller Firmware Management System Architecture Specification* requires performing of remote attestation to prove the firmware identity. The EMP firmware is required to expose firmware measurement to the host software. This measurement is represented by the cryptographic identity of all firmware components.

Two 32-byte DIGEST registers (GL_MNG_SHA_EXTEND in Section 13.2.2.29.3 and GL_MNG_SHA_EXTEND_ROM in Section 13.2.2.29.5) are implemented in the E810 for this purpose. Another GL_MNG_SHA_EXTEND_STATUS register (Section 13.2.2.29.4) exposes the measurement status.

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

Firmware flow:

1. The E810 ROM locates the E810 mini-loader.
   a. If the mini-loader is not found[1], the E810 ROM sets GL_MNG_SHA_EXTEND_ROM to the ROM version, sets GL_MNG_SHA_EXTEND_STATUS.*STAGE* to 001b, sets GL_MNG_SHA_EXTEND_STATUS.*FW_HALTED*, sets GL_MNG_SHA_EXTEND_STATUS.*DONE*, and moves to blank Flash programming.
2. If mini-loader is found, the mini-loader header contains a SHA256 hash of the mini-loader and signature over the hash. The E810 ROM authenticates mini-loader.
   a. If the authentication is successful, the E810 ROM writes the mini-loader hash value to the GL_MNG_SHA_EXTEND_ROM register and sets GL_MNG_SHA_EXTEND_STATUS.*STAGE* to 010b. In this case, the computed mini-load hash matches what is in the header, and the ROM loads the mini-loader and passes control to it.
   b. If the authentication fails, the E810 ROM writes the computed mini-loader hash value to the GL_MNG_SHA_EXTEND_ROM register, sets GL_MNG_SHA_EXTEND_STATUS.*STAGE* to 010b, sets GL_MNG_SHA_EXTEND_STATUS.*FW_HALTED*, sets GL_MNG_SHA_EXTEND_STATUS.*DONE*, and halts.
3. The E810 mini-loader locates the E810 firmware header. If the firmware is not found, the E810 mini-loader sets GL_MNG_SHA_EXTEND_STATUS.*STAGE* to 011b, sets GL_MNG_SHA_EXTEND_STATUS.*FW_HALTED*, sets GL_MNG_SHA_EXTEND_STATUS.*DONE*, and moves to blank Flash programming.
4. Firmware header contains a SHA256 hash of the firmware image. Note that the E810 mini-loader is contained within firmware. The E810 mini-loader authenticates the complete firmware image.
   a. If the authentication is successful, the E810 mini-loader writes the firmware hash value to the GL_MNG_SHA_EXTEND register. In this case, the computed firmware hash matches what is in the header. The E810 mini-loader sets the following:
      - GL_MNG_SHA_EXTEND_STATUS.*STAGE* to 100b
      - GL_MNG_SHA_EXTEND_STATUS.*DONE*
   b. If the authentication fails, the E810 mini-loader writes the computed firmware hash value to the GL_MNG_SHA_EXTEND register. The E810 mini-loader sets the following, then halts:
      - GL_MNG_SHA_EXTEND_STATUS.*STAGE* to 100b
      - GL_MNG_SHA_EXTEND_STATUS.*FW_HALTED*
      - GL_MNG_SHA_EXTEND_STATUS.*DONE*

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 might 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* indicates the latest firmware stage authenticated.

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

3.4.8.2 VPD Accesses

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 3.4.5.3 for more details.

3.4.9 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 E810 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 3.4.5 for more details. The OROM image is not authenticated by EMP on initial program load. It is authenticated only on OROM updates.

Integrity validation of NVM updates is provided by means of a digital signature. The digital signature is a SHA256 hash computed over the protected content (256 bits long) that is then encrypted by a 2048-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 decrypt the digital signature. Refer to Section 6.1.5 for more details.

To verify the authenticity of the digital signature of the mini-loader, EMP ROM must first verify that the SHA256 hash of the RSA Modulus and RSA Exponent fields in the new module loaded are identical to those retrieved from fuses. If the hash of the RSA *Modulus* and *Exponent* fields match the value in the fuses, EMP decrypts the digital signature using the 2048-bit RSA Modulus and Exponent fields stored in the manifest to extract the expected SHA256 hash of content (stored hash). EMP then performs an independent SHA256 hash over the protected content (computed hash). If the stored hash matches the computed hash, the digital signature is accepted and the NVM module update is applied.

When the mini-loader authenticates the digital signature of the full firmware or the recovery firmware, or when authentication occurs as part of a firmware update, the key hash in the authenticated firmware is compared with the one of three key hash values stored in the extended mini-loader:

- NVM Key hash (used for NVM and recovery)
- OS package key hash
- OROM key hash

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 E810's Flash by the EMP.

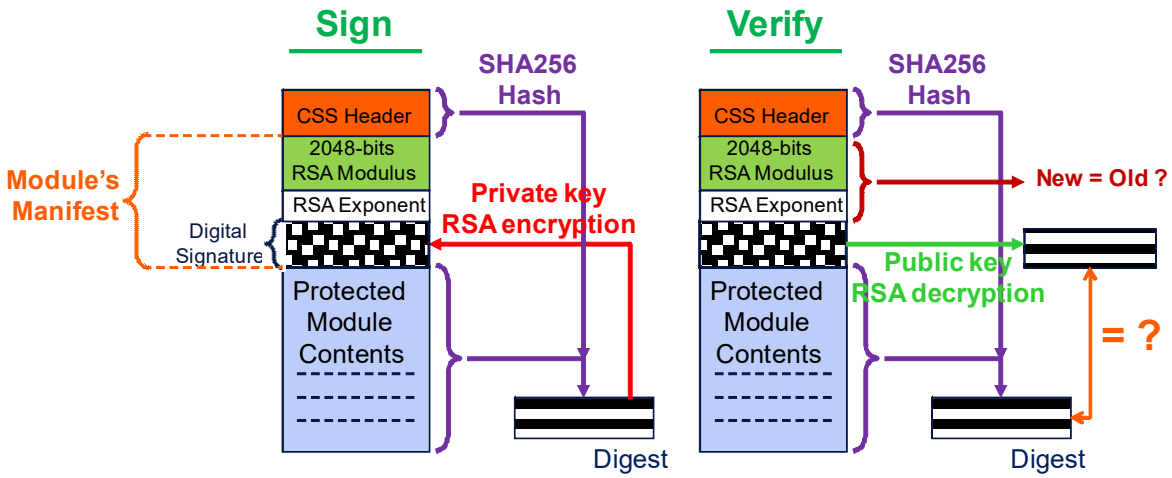


Figure 3-33. Sign and Verify Procedures for Authenticated NVM Modules

3.4.9.1 Digital Signature Algorithm Details (Not Updated with CA Certificate)

As previously mentioned, the digital signature generation is a hash computation followed by an RSA encryption. 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 E810.

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:

  http://csrc.nist.gov/publications/nistpubs/800-107/NIST-SP-800-107.pdf

- SHA validation vectors:

  http://csrc.nist.gov/groups/STM/cavp/documents/shs/SHAVS.pdf

Note: The protected module contents shown in Figure 3-33 starts with the CSS header, skips the RSA Pubic key, RSA exponent and Encrypted SHA256 hash, continues from the E810'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.

3.4.9.2 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 2048-bit RSA keys and SHA-256 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 reserved;            // Reserved for future use.
  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 previous 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 E810 CSS header includes a Security Revision (*lad_srev*) field. This field is monotonically updated for each and every security-related update to the NVM. If a security exploit is detected and an updated NVM image is released with an incremental security revision, the E810 does not allow an NVM image roll-back to an earlier version because this might expose previously known vulnerabilities. Security version checking can be temporarily disabled by manageability command.

There are four instances of the Security Revision (lad_srev) field:

- NVM module Security Revision
- Extended Mini-loader Security Revision
- Recovery Firmware Security Revision
- OROM Security Revision

The following policy applies:

- NVM module Security Revision and Recovery Firmware Security Revision should increase together.
- Extended Mini-Loader Security Revision need not increase even if others changes.
- OROM Security Revision can increase independently of the others.

Notes:

- 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.

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.

3.4.9.3 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 E810 cannot generate a signed image by itself because the private key is not available to it to generate the digital signature in the NVM.

Only the following NVM modules require authentication in the E810. Each module is with its own digital signature:

- Option ROM
- NVM bank

3.4.9.4 Software Requirements

A software tool must prepare NVM images for the CSS signing step, pre-pending the CSS manifest, and 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_FLA.*LOCKED* bit state as part of manufacturing qualification. If after the NVM is programmed, and the E810 powers up with the GLNVM_FLA.*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.

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 built into the hardware, as well as the saved CSS manifest in the NVM image.

3.4.9.5 Manufacturing Requirements

3.4.9.5.1 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 the Save Factory Settings AQC (see Section 3.4.10.9 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 the NVM Write Activate Command AQC (see Section 3.4.10.8) using preservation mode 10b.

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

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

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

3.4.9.5.2 End-of-Line Verification

OEMs must execute CELO (or an equivalent manufacturing diagnostics tool) to verify the GLNVM_FLA.*LOCKED* bit state (as described in Section 3.4.9.4).

3.4.10 NVM Access Admin Commands and Events

NVM access commands are not supported when the E810 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 9.5.13.6.

Table 3-56. NVM Access Admin Commands and Events

Command	Opcode	Description	Section Reference
NVM Read	0x0701	Read a segment from the NVM into a host buffer.	3.4.10.1
NVM Erase	0x0702	Erase consecutive 4 KB sectors of the Flash.	3.4.10.2
NVM Write	0x0703	Write a segment of the NVM from a host buffer.	3.4.10.3
NVM Config Read	0x0704	Read feature selections.	3.4.10.4
NVM Config Write	0x0705	Program Feature Selections.	3.4.10.5
NVM Checksum	0x0706	Update PFA checksum.	3.4.10.7
NVM Write Activate	0x0707	Activate new NVM.	3.4.10.8
Save Factory Settings	0x0708	Save factory settings	3.4.10.9
NVM Update EMPR	0x0709	Reset after new firmware update.	3.4.10.10
Set Package Data	0x070A	Provide package data to firmware.	3.4.10.11
Pass Component Table	0x070	Check if new module is compatible with current one.	3.4.10.12

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

3.4.10.1 NVM Read (0x0701)

This command is useful especially if the host memory-mapped access to the NVM was not enabled with the intent to save memory address space.

Table 3-57. NVM Read Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 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 19.0:** Last Command Bit Used to notify EMP that this is the last admin command of a sequence. **Bit 19.7:** 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 Module_TypeID 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 *TLV Type*). See Table 6-5 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.
Data Address High	24-27		Address of command buffer.
Data Address Low	28-31		

Table 3-58. NVM Read Response

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0701	Command opcode.
Datalen	4-5		Number of bytes the 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		*Module_TypeID* copied from the command.
Length	22-23		Length in bytes of command buffer.
Data Address High	24-27		Address of command buffer. The buffer contains the read data.
Data Address Low	28-31		

3.4.10.2 NVM Erase (0x0702)

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

Table 3-59. NVM Erase Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 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 19.0:** 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	22-31		Reserved.

This is an asynchronous command. The EMP reads the command from the ATQ 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 ATQ 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 ARQ. 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 ATQ and receiving the completion event on the ARQ.

Table 3-60. NVM Erase Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0702	Command opcode.
Datalen	4-5		Reserved.
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 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 3-61. NVM Erase Completion (on ARQ)

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 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.

3.4.10.3 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 3.4.10.2) 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 3-62. NVM Write Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 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 19.0:** Last Command Bit Used to notify EMP that this is the last admin command of a sequence. ***Note:*** Setting the *Last* bit causes a Shadow RAM dump, but it is recommended to use the NVM Write Activate to achieve this. **Bit 19.6:** Allow Special Update Set to allow change of MinSrev TLV. **All other bits** = Reserved. Must be zeroed.

Table 3-62. 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 6-5). 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 E810 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 3.4.5.4). 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 • RDMA control • HII port/PF disable • Padding TLV • Software Checksum The MinSrev module can be updated only if *Allow Special Update* is set. The following checks are applied when the MinSrev module is 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 can not 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.
Data Address High	24-27		Address of command buffer.
Data Address Low	28-31		

This is an asynchronous command. EMP reads the command from the ATQ 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 ATQ 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 ARQ. 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 ATQ and receiving the completion event on the ARQ.

Table 3-63. NVM Write Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 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 3-64. NVM Write Completion (on ARQ)

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 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 19.0:** A reset flag to indicate the type of reset required to get the NVM bank update effective. 0b = POR 1b = PERST **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		

3.4.10.4 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 3-65. NVM Config Read Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 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 16.0:** Single/Multiple elements. 0b = Only a single *Feature_ID*/*Field_ID* is read. 1b = *Feature_ID*/*Field_ID* iteration is used. **Bit 16.1:** 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 *Feature_ID*/*Field_ID* when Bit 16.0 is 0b. *Feature_ID* when Bit 16.1 is 0b. *Field_ID* when Bit 16.1 is 1b. Iterator: *Feature_ID*/*Field_ID* to start reading from (iterator) when Bit 16.0 is 1b. ***Note:*** When Bit 16.0 is 0b, *Feature_ID*/*Field_ID* = 0 is not valid. When Bit 16.0 is 1b, it indicates that the command reads the data starting from the first *Feature_ID*/*Field_ID* in the array.
Reserved	22-23		Reserved.
Data Address High	24-27		Address of command buffer.
Data Address Low	28-31		

Table 3-66. NVM Config Read Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 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 16.0:** Single/Multiple elements. 0b = Only a single *Feature_ID*/*Field_ID* is read. 1b = *Feature_ID*/*Field_ID* iteration is used. **Bit 16.1:** 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	22-23		Reserved.
Data Address High	24-27		Address of command buffer. See Section 3.4.10.6.
Data Address Low	28-31		

3.4.10.5 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 3-67. NVM Config Write Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 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 16.0:** Reserved. **Bit 16.1:** Feature/Field 0b = Feature selections are written. 1b = Immediate fields are written. **Bit 16.2:** 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.
Data Address High	24-27		Address of command buffer.
Data Address Low	28-31		

Table 3-68. NVM Config Write Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 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.

3.4.10.6 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 3-69. 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 3-70. 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.

3.4.10.7 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 3-71. NVM Update Checksum Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 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 17.0:** Verify Checksum **Bit 17.1:** 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 3-72. NVM Update Checksum Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 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 *Verify Checksum* flag was set in command.
Reserved	20-31		Reserved.

3.4.10.8 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 3-73. NVM Write Activate Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 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 18.0:** 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 19.0:** Reserved. **Bits 19.2-19.1:** 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 **Bit 19.3:** 0b = Keep current NVM Bank. 1b = Switch to invalid NVM Bank **Bit 19.4:** 0b = Keep current NV OROM Bank. 1b = Switch to invalid OROM Bank **Bit 19.5:** 0b = Keep current EXT TLV Bank. 1b = Switch to invalid EXT TLV Bank **Bit 19.6:** 0b = No event. 1b = Revert Last Activate. **Bit 19.7:** 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 3-74. NVM Write Activate Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 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). EPERM = 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 3-75. NVM Write Activate Completion (on ARQ)

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 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.

3.4.10.9 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 3.4.9.5.1, "Factory Settings Preservation Flow".

Table 3-76. Save Factory Settings Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 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 3-77. Save Factory Settings Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 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.

3.4.10.10 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 3-78. NVM Update EMPR Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 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 3-79. NVM Update EMPR Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 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. EPERM = 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.

3.4.10.11 Set Package Data (0x070A)

This command is equivalent to the reception of a PLDM FW 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 3-80. NVM Set Package Data Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 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 19.0:** 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	20-23		Reserved.
Data Address High	24-27		Address of command buffer. The buffer contains the PackageData.
Data Address Low	28-31		

Table 3-81. NVM Set Package Data Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 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 19.0:** 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	20-23		Reserved.

Table 3-81. NVM Set Package Data Response [continued]

Name	Byte.Bit	Value	Remarks
Data Address High	24-27		Address of command buffer.
Data Address Low	28-31		

3.4.10.11.1 Buffer of 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].Bits [7:1] — 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 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.

3.4.10.12 Pass Component Table (0x070B)

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 E810 assumes these commands are going to be sent until the *TransferFlag* is set to *End* or *StartAndEnd*.

Table 3-82. NVM Pass Component Table Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 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 *TransferFlag* is ignored by firmware for now.
Reserved	20-23		Reserved.
Data Address High	24-27		Address of command buffer. Contains the component table.
Data Address Low	28-31		

Table 3-83. NVM Pass Component Table Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x070B	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). 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.

Table 3-83. NVM Pass Component Table Response [continued]

Name	Byte.Bit	Value	Remarks
ComponentResponse	16		*ComponentResponse* as described in DSP0267: 0 = Component can be updated - *ComponentResponseCode* must be set to 0x00. 1 = Component might not be updatable - A *ComponentResponseCode* 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 *UpdateOptionFlags* field within the *UpdateComponent* 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 *ComponentResponseCodes* are denoted below. All other values reserved. This value reflects the update status this module.
ComponentResponseCode	17		*ComponentResponseCode* as defined in DSP0267 and as described in Section 12.8.4.2.4: 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 *ComponentResponse* = 2. Returned, among others, if GFID does not match. 0x04 = Component has conflict with another component provided in a separate *PassComponentTable* 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 *ComponentResponse* = 2. 0x06 = Component is not supported on FD. The device rejects the request and returns a *ComponentResponse* = 2. 0x07 = Security restrictions prevent component from being downgraded. Should include Srev check. The device rejects the request and returns a *ComponentResponse* = 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.
Reserved	18-23		Reserved.
Data Address High	24-27		Address of command buffer.
Data Address Low	28-31		

3.4.10.12.1 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 3-84. For more information, see Section 12.8.5

Table 3-84. Component

Type	Name	Format/Value
uint16	**ComponentClassification**	Should be 0x000A - Firmware for all components.
uint16	**ComponentIdentifier**	NVM = 0x6 OROM = 0x5 Topology Net List = 0x8
uint8	**ComponentClassificationIndex**	0x0 - Not used.
uint32	**ComponentComparisonStamp**	MajorMinor Major = 16 bit Minor = 16 bit NVM bank comparison stamp = 31:25: DevStarterVersion Major (0-99) 24:18: DevStarterVersion Minor (0-99) 17:0: EETrackID OROM bank comparison stamp = CIVD if Combo - Combo Image Version Low/High in CIVD Section. In case of no OROM, the version reported is all zeros. Net list comparison stamp = BaseReleaseVersion.Major/Minor (32 bit value copied from Netlist identifier block).
enum8	**ComponentVersionStringType**	ASCII = 1
uint8	**ComponentClassificationIndex**	The length, in bytes, of the ComponentVersionString.
Variable	**ComponentVersionString**	See section Section 12.8.5.2.1.

3.4.11 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 3.4.5.3.1.

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 Table 3-85 and Table 3-86.

Table 3-85. Small Resource Structure

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

Table 3-86. Large Resource Structure

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

The E810 parses the VPD structure during the auto-load process following PCIe reset to detect the read-only and read/write area boundaries. The E810 assumes the following VPD fields with the limitations listed:

Table 3-87. 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 E810 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 3-87, 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 3-87. 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 and 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 14.3.4). 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 3.4.5.3 for more details.

3.5 General Purpose I/O (GPIO) and LED

The E810 provides two types of GPIOs:

- Low latency GPIOs, that are directly controlled by the device registers.
- Standard GPIOs that are provided by an I/O Widget.

Four direct, low latency GPIOs are connected directly to single-ended 3.3 V IO pads, plus one differential output. In addition to the direct low latency GPIOs, the E810 controls an I/O Widget that provides an additional 20 GPIOs, which can be used either as SDPs or LEDs.

In total the E810 provides the following:

- Four direct low latency single-ended GPIOs, plus one differential output, mostly used for TimeSync application.
- An additional 20 GPIOs, controlled using an I/O Widget. The I/O Widget IP is able to provide up to 32 general-purpose GPIO pins. However, the following SDP/LED are externally connected.
  - 8 SDP pins, mostly for allowing direct control of SFP modules (for example, the MOD_ABS pin).
  - 12 LED pins that can be divided into 3 LEDs per port in 4 SFP+ configuration.

Note: The I/O Widget generates LEDs and SDPs with the same feature set, so the division to 12 LEDs and 8 SDPs is provided only as an example to typical low port count application

3.5.1 E810 I/O Widget SDP and LED

The E810 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 (such 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 be 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 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 I/O Widget SDP and LEDs. See Section 3.3.

3.5.1.1 E810 GPIO Pin Names

Table 3-88 summarize the GPIO pin names:

Table 3-88. E810 GPIO Pins

Pin Name	Source	Signal Name	Control Register	Usage
SDP0	I/O Widget	iow_gpio_o/i[0]	IOW/GPIO_CTL[0]	LED or SDP assigned via the link topology netlist.
SDP1	I/O Widget	iow_gpio_o/i[1]	IOW/GPIO_CTL[1]	
SDP2	I/O Widget	iow_gpio_o/i[2]	IOW/GPIO_CTL[2]	
SDP3	I/O Widget	iow_gpio_o/i[3]	IOW/GPIO_CTL[3]	
SDP4	I/O Widget	iow_gpio_o/i[4]	IOW/GPIO_CTL[4]	
SDP5	I/O Widget	iow_gpio_o/i[5]	IOW/GPIO_CTL[5]	
SDP6	I/O Widget	iow_gpio_o/i[6]	IOW/GPIO_CTL[6]	
SDP7	I/O Widget	iow_gpio_o/i[7]	IOW/GPIO_CTL[7]	
SDP8	I/O Widget	iow_gpio_o/i[8]	IOW/GPIO_CTL[8]	
SDP9	I/O Widget	iow_gpio_o/i[9]	IOW/GPIO_CTL[9]	
SDP10	I/O Widget	iow_gpio_o/i[10]	IOW/GPIO_CTL[10]	
SDP11	I/O Widget	iow_gpio_o/i[11]	IOW/GPIO_CTL[11]	
SDP12	I/O Widget	iow_gpio_o/i[12]	IOW/GPIO_CTL[12]	
SDP13	I/O Widget	iow_gpio_o/i[13]	IOW/GPIO_CTL[13]	
SDP14	I/O Widget	iow_gpio_o/i[14]	IOW/GPIO_CTL[14]	
SDP15	I/O Widget	iow_gpio_o/i[15]	IOW/GPIO_CTL[15]	
SDP16	I/O Widget	iow_gpio_o/i[16]	IOW/GPIO_CTL[16]	
SDP17	I/O Widget	iow_gpio_o/i[17]	IOW/GPIO_CTL[17]	
SDP18	I/O Widget	iow_gpio_o/i[18]	IOW/GPIO_CTL[18]	
SDP19	I/O Widget	iow_gpio_o/i[19]	IOW/GPIO_CTL[19]	
SDP20	Controller Core	gbe_o/i_gpio[0]	GLGEN_GPIO_CTL[0]	1588 timing support.
SDP21	Controller Core	gbe_o/i_gpio[1]	GLGEN_GPIO_CTL[1]	
SDP22	Controller Core	gbe_o/i_gpio[2]	GLGEN_GPIO_CTL[2]	
SDP23	Controller Core	gbe_o/i_gpio[3]	GLGEN_GPIO_CTL[3]	
CLK_OUT_p/n	Controller Core	gbe_o/i_gpio[5]	GLGEN_GPIO_CTL[5]	

Chapter 4 Initialization

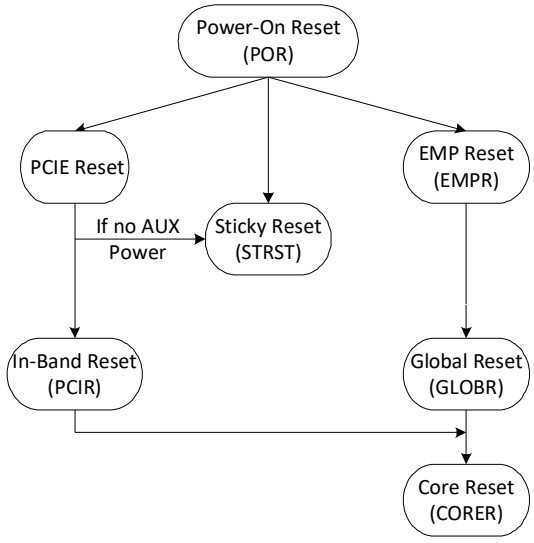
4.1 Reset Operation

The following sections lists the hardware and software reset sources that initialize the entire portions of the E810 and function-level resets. The reset sources are listed in Section 4.1.1 and the reset flows are detailed in Section 4.1.2.

4.1.1 Reset Sources

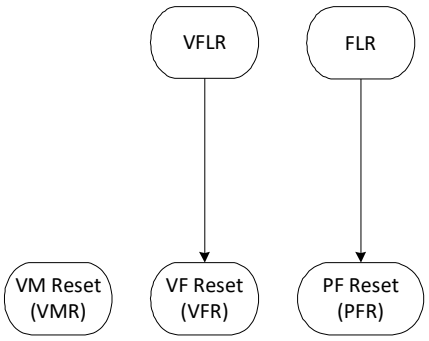
This section lists the reset sources supported by the E810, while the complete list of initialized logic is listed in Table 4-1.

A hierarchical reset tree is shown in the Figure 4-1. Any logic initialized by a specific reset is initialized also by any other reset source that is linked to it and located above it in the reset tree.



Reset Source	Cause List of the Reset Source
Power-On Reset (POR)	Power applied to the device.
Sticky Reset (STRST)	Internal reset that clears PCIe sticky registers following PCIe reset.
PCIE Reset	IOSF Primary/secondary reset input signal.
In-Band Reset (PCIR)	In-Band PCIe message signaling.
EMP Reset (EMPR)	EMP watchdog timer expiration or Force TCO command.
Global Reset (GLOBR)	Device driver setting the GLOBR bit in the GLGEN_RTRIG register.
Core Reset (CORER)	Device driver setting the CORER bit in the GLGEN_RTRIG register.

Figure 4-1. Hierarchical Hardware Reset Tree



Reset Source	Cause List of the Reset Source
PF Reset (PFR)	FLR assertion via the PCIe configuration space. D0 to D3 transition or Clearing the BME. Device driver setting the PFSWR bit in the PFGEN_CTRL register.
VM Reset (VMR)	Device driver setting the VMR bit in the VSIGEN_RTRIG register.
VF Reset (VFR)	VFLR assertion or setting the VFSWR bit in the GLGEN_VFRTRIG registe.r
PF Level reset (FLR)	Operating system sets the FLR bit in the PF PCIe configuration.
VF Level reset (VFLR)	Operating system sets the FLR bit in the VF PCIe configuration or clearing the VFE bit in the parent s PF PCIe configuration.

Figure 4-2. Function Virtual Resets

Hardware Resets:

- **Power-On Reset (POR)** — A full power-on flow is translated to sideband reset and global reset signals to the E810.
- **PCIe Reset (primary reset de-assertion)** — 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 as listed in Table 4-1. De-assertion of primary reset also triggers internally a PCIR, which is detailed later in this section. Global reset is also de-asserted if link is not needed for APM and manageability is disabled.
- **Sticky Reset (STRST)** — Sticky reset is internal signal triggered by PCIe reset when the E810 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.
- **In-Band Reset (PCIR)** — PCIe supports in-band signaling for PCIe reset (called PCIR). Any cycles on the PCIe bus are gated instantly as well and packet transmission generated by software. The entire data path is initialized other than the EMP cluster. Although the EMP subsystem is not initialized, some packets of the EMP might be lost during a short window of about 1 µs.
- **Core Reset (CORER)** — CORER initializes the shared data path for all functions excluding the EMP subsystem, PCI interface, and MAC/PHY logic of all ports. Any further master cycles of all PFs and VFs are not initiated while some packets that were already fetched completely might still be sent out. Even though the EMP subsystem is not cleared, pass-through traffic might be inhibited during the initialization cycle that might take ~20 ms. Also, SMBus accesses are responded with NACK during the initialization cycle. This reset is not expected to be used other than as an escape mechanism in case the E810 hangs and the PFR did not resolve the problem. This reset is initiated by the PFs by setting the *CORER* bit in the GLGEN_RTRIG register. The EMP initiates this reset by setting a bit in an internal register. As seen in Figure 4-1, this reset is also set as part of EMPR reset or Global reset (higher level reset).
- **Global Reset (GLOBR)** — GLOBR is a superset CORER initializing any logic initialized by the CORER, plus the MAC/PHY logic of all ports (both internal PHY and external PHY if connected). This reset is not expected to be used other than escape mechanism in case CORER did not resolve the problem. This reset is initiated by the PFs by setting the GLOBR bit in the GLGEN_RTRIG register. The EMP initiates this reset by setting a bit in an internal register. Global reset is also initiated following a Force TCO command (if it is not disabled by the Force TCO Reset Disable bit in the NVM). As seen in Figure 4-1, this reset is also set as part of EMPR reset (higher level reset) and this reset also generates CORE reset.

- **EMP Reset (EMPR)** — EMPR initializes the resources and data path connected to the EMP including its firmware reload (excluding internal-SR). EMPR is triggered internally by the EMP watchdog timer expiration or by EMP setting an internal flag or due to Force TCO command or due to uncorrectable ECC error in one of the EMP memory shells. As seen in Figure 4-1, this reset generates CORE reset and global reset.

Function-Level Resets:

- **Function Level Reset (FLR)** — The E810 supports the standard FLR interface in the Device Control register of the PCIe capability structure within the PCI configuration space of the PFs. Setting the *FLR* bit initializes the PCI configuration of the PF (including the *VFE* flag in the SR-IOV Control/ Status register that disables all the VFs of the PF) and initiates an internal PFR described later in this section.
- **PF Reset (PFR)** — PFR initializes the resources and data path of the PF and its VFs with no impact on other PFs, VFs, or the EMP subsystem. Any further master cycles of the PF are not initiated while some packets that were already fetched completely might still be sent out. The PFR is generated by one of the following four causes:
  - D0 to D3hot transition, which is also known as ACPI reset.
  - FLR
  - PF software sets the *PFSWR* bit in the PFGEN_CTRL register.
  - De-assertion of the *Bus Master Enable* flag in the PCI configuration space.
- **VF Level Reset (VFLR)** — The E810 supports the standard VFLR interface in the Device Control register of the PCIe capability structure within the PCI configuration space of the VFs. Setting the *VFLR* bit initializes the PCI configuration of the VF and initiates an interrupt to the PF that completes the VFR reset described later in this section. Clearing the *VFE* flag in the PF configuration space also impacts all its VFs the same as a VFLR.
- **VF Reset (VFR)** — VFR initializes the resources and data path of the VF with no impact on other PFs, VFs, or the EMP subsystem. Any further master cycles of the VF are not initiated, while some packets that were already fetched completely might still be sent out. The VFR is generated by one of the following two causes:
  - VFLR or clearing the *VFE* bit of the parent PF. Note that after the VFE bit is cleared, the PF driver should follow the VFR flow for all the VFs of the PF, including those VFs that are not enabled.
  - PF software sets the VFSWR bit in the VPGEN_VFRTRIG register of its VF.
- **VM Reset (VMR)** — There are 768 VSIs. VMR is a mechanism to reset a VSI. VMR initializes the resources and data path of the VM with no impact on other PFs, VFs, VMs, or the EMP subsystem. Any further master cycles of the VM are not initiated, while some packets that were already fetched completely might still be sent out. The VMR is generated by the PF software setting the *VMSWR* bit in the VSIGEN_RTRIG register.

Table 4-1. Device Logic Affected by the Reset Sources

Reset Activation	POR	STRST / PCIR / PERST	EMPR	GLOBR	CORER	FPR / FLR	VMR	VFR/ VFLR
Load the NVM to the Shadow RAM in the hardware and clear the alternate structure	+							
Load the EMP and PE firmware from the NVM	Both+	PE	Both+	PE	PE			
Load device settings from NVM shadow and alternate structure (see details listed in Table 6-2 and the reset flow sections that follow)	Shad	Both	Both	Both	Both			
PF MAC Addresses (switch and WoL)[1]	+	+[1]	+	+	+	+		
MAC and PHY interface	+	[1]	+	+				
EMP subsystem including firmware reload	+		+					
Sticky PCIe context	+	[2]						
PCIe HWInit parameters	+	PERST only						
PCIe RO registers	+	+						
PCIe RW/RW1C registers	+	+				+[3]		VF by VFLR
Bus master disable[4]	+	+	+	+	+	PF	VM	VF
All CSRs (Refer to Section 13, "Programming Interface" for the reset source of each register.)								
Most of the PF and VF registers (PF registers are named "PFxxx", and VF registers named "VFxxx" and "VPxxx"). Refer to Section 13, "Programming Interface" for the reset source of each register.	+	+	+	+	+	PF and its VFs		VF only
Cache contexts (Quad hash filters, PE context) and FPMs settings (sector descriptors and cache entries)	+	+	+	+	+	PF and its VFs	VM	VF
Invalidate VF queue mapping tables (VPLAN_QTABLE)	+	+	+	+	+	VFs of the PF		VF
Invalidate VSI context (including the switch, VSILAN_QTABLE and all other VSI registers)	+	+	+	+	+	PF and its VFs	By the PF SW[5]	By the PF SW[5]
Load the PFs MAC Address to the switch and the WOL filters from the NVM (not all PFs must have a WOL filter)	+	+[6]	+	+	+	PF		
Tx and Rx data path + Tx-Scheduler	+	+	+	+	+	PF and its VFs	VM	VF
Tx and Rx packet buffers	+	+	+	+	+			
Tx and Rx queue disable	+	+	+	+	+	PF and its VFs	By the PF SW	VFs (Done by FW for Rx queues)
Admin queue disable (POR and EMP also clear the queue context memories.)	+	+	+	+	+	PF and its VFs		VF
Disable interrupts	+	+	+	+	+	PF and its VFs		VF
Interrupt cause control registers	+	+	+	+	+	By the PF SW	By the PF SW	By the PF SW

Table 4-1. Device Logic Affected by the Reset Sources [continued]

Reset Activation	POR	STRST / PCIR / PERST	EMPR	GLOBR	CORER	FPR / FLR	VMR	VFR/ VFLR
RSS key and table	+	+	+	+	+	PF		VF
Invalidate FD filters	+	+	+	+	+	PF	VM (FW clears all filters on VSI clear)	VF
Invalidate all internal caches: transmit and receive queue contexts; PE Quad Hash.	+	+	+	+	+			
Invalidate the FPM tables of the PF and its VFs (function private memory).	+	+	+	+	+	PF and its VFs		VF

1. PF MAC Addresses (switch and WoL) are cleared by hardware at POR and loaded from NVM by the firmware at any of the highlighted reset causes. Following PCIR, the switch MAC Addresses are loaded following the assertion of the PCIe reset or in-band reset, and the WoL MAC Addresses are loaded only following the de-assertion of the PCIe reset.The MAC and PHY are cleared only if link is not needed for APM and MNG is disabled. GLOBR if link is not needed for APM and MNG is disabled. If the WUC filter is inactive and no port is needed by firmware/MNG (PRTPM_GC.*EMP_LINK_ON* is clear), GLOBR is maintained low by hardware during PCIR de-assertion for power savings.
2. Sticky PCIe context is cleared on PERST if no AUX power as documented in Section 14.2.1).
3. For exception list of registers that are not cleared on PFR, see Section 14.2.1.
4. The E810 has several flags that control the Bus Master Enable (BME). BME flags on the PCI configuration space (for each PF and VF) and the VMRD flag in the VSIGEN_RSTAT registers for each VM that is not a VF. The BME flags of all PFs are cleared by all reset causes indicated by + symbol and a specific PF by FLR. The BME flags of all VFs are cleared by all reset causes indicated by + symbol and a specific VF by VFLR. The VMRD flags of all VSIs are set by all reset causes indicated by + symbol and a specific VSI by VMR. Note that as opposed to BMEs of the function that are cleared by the previous resets (disabling master accesses), the VMRD flags are set (enabling bus master accesses when the BME of their parent PF is set as well).
5. VSIs and its related switch context are cleared by admin command(s) initiated by the software.
6. Note that as opposed to any logic in the E810 that is initialized at the leading edge of PERST#, the WOL filters are initialized at the de-assertion of PERST# (entering the D0u state).

Table 4-2. NVM Section Loaded by Reset Source Covered by GLNVM_ULD Register

NVM Section	POR	PERST	PCIR	EMPR	GLOBR	CORER
POR Registers Auto-Load	+					
PCIe Analog Configuration	+					
PCIR Registers Auto-Load	+	+	+			
PCIe Transaction Layer (TL) Shared	+	+	+			
CORER Registers Auto-Load	+	+	+	+	+	+
GLOBR Registers Auto-Load	+			+	+	
PHY Analog Configuration	+					
EMPR Registers Auto-Load	+			+		

4.1.2 Hardware Reset Flows

This section describes the reset flows in the E810 and software interaction.

4.1.2.1 POR Flow

Power-up sequence described in this document starts at power-on reset towards the E810.

1. Power-on reset is de-asserted. Crystal clocks are stable and toggling, and memory repair process has completed.
2. IPs perform fuse polling. At that stage, the SPI controller is ready to accept messages from other IPs.
3. The E810 starts its boot process, checks the validity of the NVM, and loads the initial firmware code (mini-loader). The mini-loader code is authenticated by the E810.

   Note: PCI reset de-assertion occurs at least 100 ms after power-up flow is started. Since flow duration is less than 100 ms, it is guaranteed that firmware is ready for PCI reset de-assertion at the end of this stage.

   a. E810 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).
4. First load process. E810 firmware loads:
   a. HIP's PLL configuration to the HIP modules. HIP's auto-loaded data is pushed over SB-IOSF by the E810.
   b. CSR protected list.
   c. POR section from Shadow RAM.
5. E810 firmware loads and authenticates the full firmware code content.
6. Second load process:
   a. E810 firmware loads the full HIP configuration.

Figure 4-3 shows the stages through the power-up sequence. The numbers relate to the steps previously described.

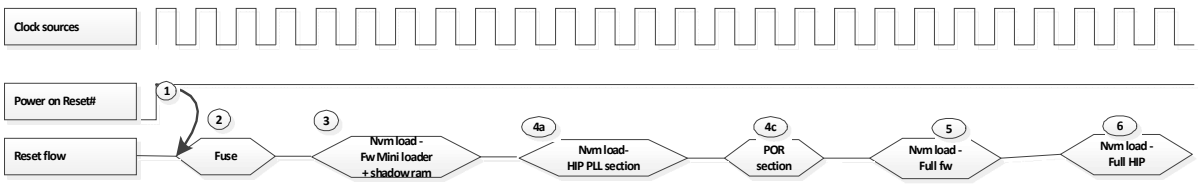


Figure 4-3. External Power-On Flow

Table 4-3 lists the timing of each of the steps.

Table 4-3. External Power-Up Sequence

Step		Duration (ms)[1]	Completion Time from Start (ms)[2]
1	Power-on reset.	0	40
2	Read fuse and straps information.	4	44
3	Load mini-loader from the NVM[3] + Load Shadow RAM from the NVM.	20	64
4	Initial configuration of hardware blocks (auto-load).		

1. The 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.

4.1.2.2 Primary Reset and In-Band PCI Reset Flow

The internal reset flow is shown in Figure 4-4 and described in this section.

Note: Core reset might be extended if none of the functions are enabled for WoL or Path-through.

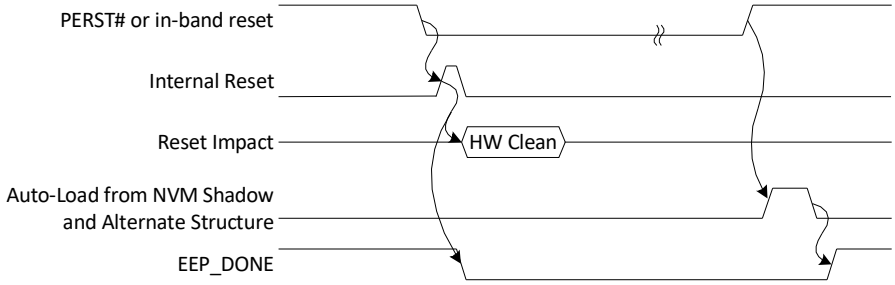


Figure 4-4. PCI Reset Flow

Hardware response:

1. Avoid any further master accesses on the PCIe bus and discard any completions for the PF.
2. Reset PCIe registers and core registers as listed in Table 4-1.
3. The following flags are cleared by the E810 when:
   - The *CONF_*_DONE* bits in GLNVM_ULD register are cleared.
4. The E810 loads the NVM section into hardware. After loading, the EMP might start responding to the Get Version Admin command, which is blocked until this stage.
   - When checking for hardware configuration to complete, software device driver should either wait for a response to Get Version Admin command, or poll on the *CONF_*_DONE* bits.

Following these steps, hardware is ready to accept operating system configuration cycles.

Note: EMP packets might be lost as well in a window frame of tens of milliseconds.

4.1.2.3 Core, Global, and EMP Reset Flows

The global resets can be initiated by the PF software or the EMP firmware. It is expected to be used as a mechanism to resolve potential hardware locks or synchronization lose, bringing the E810 to a known functional state. These global resets impact all PFs (and their VFs) as well as the EMP subsystem (EMP reset only). Therefore, graceful flow is enabled by hardware as shown in Figure 4-5 and described in the following sections. The software initiates the global resets by the GLGEN_RTRIG register. The EMP can access the same register or use an internal register.

As seen in the "Hierarchical Reset Tree" diagram (Figure 4-1):

- EMPR reset also triggers a global reset and a core reset.
- Global reset can be self asserted. When asserted, it also triggers a core reset.
- Core reset can be self-asserted.

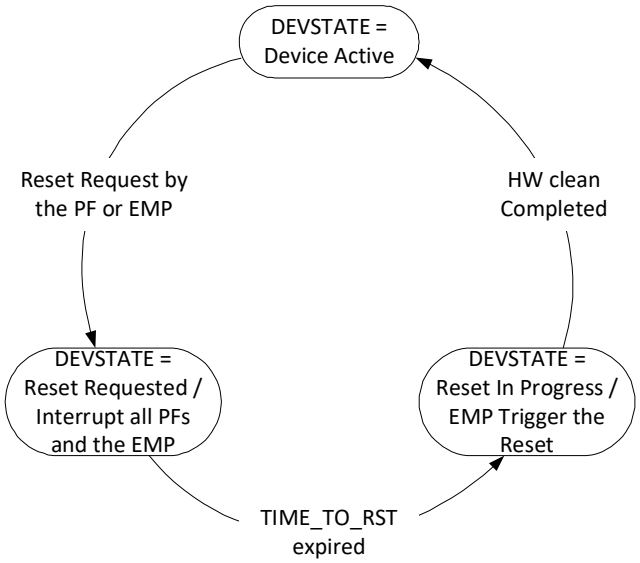


Figure 4-5. Global Reset Flow

4.1.2.3.1 Software and Firmware Interface

- **GLOBR/CORER** — Setting the *GLOBR* flag or *CORER* flag in the GLGEN_RTRIG register triggers a graceful global/core reset, respectively.
- **EMPFWR** — EMP firmware watchdog expiration or EMP setting internal EMP reset flag, triggers a graceful EMP reset (EMPR).
- **DEVSTATE** — Global device state exposed to all PFs in the GLGEN_RSTAT registers. The DEVSTATE can be at one of the following states: device active, reset requested, or reset in progress.
- **RESET_TYPE** — The RESET_TYPE reflects the last reset cause initiated to the E810. It changes its state once any of the following reset sources is triggered. The *RESET_TYPE* can be at one of the following states: POR, CORER, GLOBR, or EMPR.
- **RST_CNT** — The GLGEN_RSTAT reflects the following counters: CORERCNT, GLOBRCNT, and EMPRCNT. These fields are each 2-bit counters. They count the matched reset completion events since POR.

4.1.2.3.2 CORER Flow

The PF software or EMP firmware initiates a graceful core reset by setting the *CORER* flag in the GLGEN_RTRIG register and polling the E810 state in the GLGEN_RSTAT register.

Note: If a CORER occurs while internal Shadow RAM is inconsistent with Flash Shadow RAM content, an EMPR is triggered as part of the CORER flow.

Hardware/EMP firmware response:

1. Changes to the GLGEN_RSTAT register are as follows:
   a. Set *DEVSTATE* to reset requested.
   b. Set *RESET_TYPE* to CORER indicating the requested reset.
   c. Initiate a GRST interrupt to all PFs and EMP that the reset is about to be fired by hardware.
2. As a response to interrupt, EMP reads *TIME_TO_RST* by the GLGEN_RSTCTL.*GRSTDEL* value and start counting down. When timer expires, EMP triggers the reset via the GLGEN_IMRTRIG register. The *CONF_*_DONE* bits in GLNVM_ULD are cleared by the E810 when reset is asserted.
3. Clear the entire transmit and receive data path and core registers as listed in Table 4-1.
4. Avoid any further master accesses on the PCIe bus and discard any completions for the entire device, and abort any transmission in progress (including packets sent by the EMP).
5. Reset internal device logic excluding EMP cluster, PCI interface, and MAC/PHY cluster as listed in Table 4-1.
6. Once the reset flow is completed, update the GLGEN_RSTAT register as follows:
   a. Set *DEVSTATE* to device active.
   b. Increment the *CORERCNT* by one.
7. The E810 loads the NVM section into hardware. After loading, the EMP might start responding to the Get Version Admin command, which is blocked until this stage.

Following the GRST interrupt, all PFs poll the E810 state in the GLGEN_RSTAT register.

1. Keep polling the register as long as DEVSTATE is not equal to device active. The GLGEN_RSTAT register also indicates the number of the initiated global resets via the GLGEN_RTRIG register (CORER, GLOBR, EMPR). These counters might be needed in case a function initiated consecutive global reset before all other functions had the chance to realized that the previous reset was completed. It is a good practice to avoid too frequent global resets to avoid such cases.
2. Check the *RESET_TYPE* that indicates the reset that was initiated and follow the required initialization flow.

   Note: Also, the function that initiated the reset should check the *RESET_TYPE*, since it might reflect a stronger reset initiated by another function.
3. Check that the hardware completed to auto-load its settings from the NVM shadow and the alternate structure by polling the *CONF_*_DONE* flags in GLNVM_ULD register.

   Note: When checking for the EMP configuration to be done, software should either wait for a response to a Get Version Admin command or poll on the *CONF_*_DONE* bits.
4. The PFs proceeds with their software initialization flow.

4.1.2.3.3 GLOBR Flow

Global reset is initiated by the PFs or the EMP by setting the *GLOBR* flag in the GLGEN_RTRIG.

As seen in the "Hierarchical Reset Tree" diagram (Figure 4-1),global reset also triggers core reset.

The GLOBR flow is identical to the CORER flow with the following changes:

- GLOBR also initializes the MAC/PHY units.
- GLGEN_RSTAT.*RESET_TYPE* is set by the hardware to GLOBR (rather than CORER).
- Increment the *GLOBRCNT* by one (rather than *CORERCNT*).
- Loading the E810 setting from the NVM is listed by the GLOBR column in Table 4-2.

Note: If a GLOBR occurs while internal Shadow RAM is inconsistent with Flash Shadow RAM content, an EMPR is triggered as part of the CORER flow.

4.1.2.3.4 EMPR Flow

EMP reset is expected to be used by the EMP as a mechanism to resolve potential hardware locks or potential loss of synchronization between the firmware and hardware that are not expected to be resolved by CORER nor by GLOBR. The EMPR impacts also all PFs and their VFs. Therefore, the following graceful flow is recommended.

As seen in the "Hierarchical Reset Tree" diagram (Figure 4-1) EMPR reset also triggers and global reset and a core reset.

The EMP flow is identical to the CORER flow with the following changes:

- During normal operation, the EMPR can be initiated only by the EMP by setting an internal EMP reset flag.
- EMPR initializes also the MAC/PHY units as well as the EMP cluster.
- GLGEN_RSTAT.*RESET_TYPE* is set by hardware to EMPR (rather than CORER).
- Increment the *EMPRCNT* by one (rather than *CORERCNT*).
- Loading the E810 setting from the NVM is listed by the EMPR column in Table 4-2.

4.1.3 Function-Level Reset Flows

VF/VM/PF reset requests (VPGEN_VFRTRIG for VF reset, VSIGEN_RTRIG for VM reset, and PFGEN_CTRL.*PFSWR* for PF reset) are ignored in the following cases:

- CORER reset is asserted (also during pre-indication of CORER)
- Between CORER de-assertion and CORER NVM load completion (fleep_al_corer_done set)

Note: During a function-level reset, spurious malicious events might occur due to anti-spoof.

4.1.3.1 PFR Flow

PFR resets a specific PF. For SR-IOV, it also resets the VFs of this PF. The reset flow is shown in Figure 4-6. It is initiated by the PF driver (setting the *PFSWR* bit in the PFGEN_CTRL register) or operating system (setting the *FLR* flag in the Device Control register) or D0 to D3hot transition.

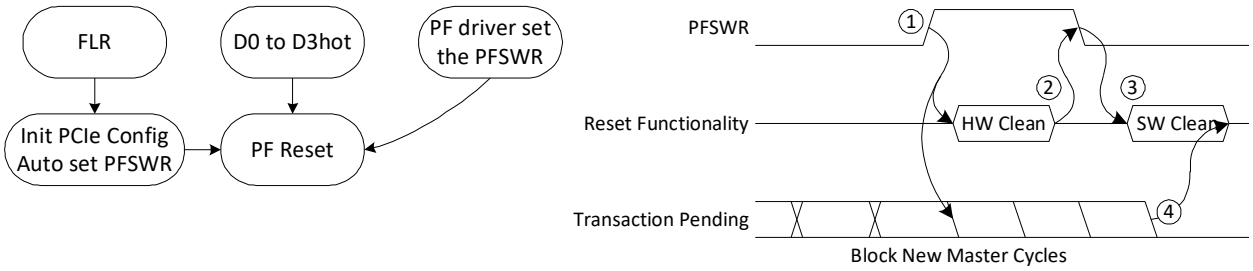


Figure 4-6. PFR Triggers

The PF hardware response to PFR is listed in Table 4-1.

The E810 hardware initiates two orthogonal flows as a response to FLR:

PCIe clean-up flow:

1. Emulate internal VFR to all its VFs (the hardware response to VFR is described in Section 4.1.3.3.3 with the exception that the VF's CSRs are not gated on read).
2. Avoid any further master accesses on the PCIe bus and discard any completions for the PF and its VFs.
   a. Once all pending requests of the PF are completed, the *Transaction Pending* bit in the Device Status register in the PCIe configuration space is cleared.
   b. Once all pending requests of each VF of the PF are completed, the *Transaction Pending* bit in the Device Status register the VF PCIe configuration space is cleared (per each VF).

Data path clean up flow:

1. Upon event, EMP does the next "pre hardware drain" clean ups:
   a. Cleans up switch population of the PF.
   b. Cleans up ACL population of the PF (as described in Section 7.9.2.10).
   c. Cleans up RSS population of the PF (as described in Section 7.10.11.2).
   d. Cleans and disables FD filters of the PF (as described in Section 7.10.12).
   e. Disables mailbox queues and sideband queues of the PF and its VFs in SR-IOV mode, or its VMs when working in PASID mode (as described in Section 9.5.3).
   f. Clears the enable bit of doorbell queues of the PF (doorbell queue context layout is described in Section 10.5.5.6).
   g. Halts all the initialized transmit queues using the following flow (firmware must not halt an uninitialized queue):
      1. Set count down register GL_TCVMLR_QCNTR to the number of disabled queues
      2. Per each halted Tx-Queue, EMP sends queue halt command using GL_TCVMLR_QCTL register. Queue IDs are described in the global space.
      3. Tracks the completion of the multiple Q halt commands using GL_TCVMLR_QCNTR register. The entire Q halt operation is completed when this register is zeroed. Firmware can load value to this register only when it is equal to zero. Must not initiate any Q halt flow before the previous one is completed.

2. Via the GL_XLR_MARKER_TRIG* registers, EMP indicates to hardware to send a drain marker that drains the Tx data path, Rx data path, and PE data path orthogonally.
   - The specific GL_XLR_MARKER_TRIG* registers are:
     — GL_XLR_MARKER_TRIG_VMLR
     — GL_XLR_MARKER_TRIG_TCVMLR
     — GL_XLR_MARKER_TRIG_RCU_PRS
     — GL_XLR_MARKER_TRIG_PE (should be written only if PW is enabled).
   - Draining the Tx and Rx data paths might take some time when the TC is paused by flow control. Hardware includes a timeout mechanism that prevents indefinite latency, as described in Section 8.2.4.4.1.
   - As the second part of the data path clean up flow (cache clean-up phase after main data path is drained), PE firmware, cleans and disables QH filters of the PF (as described in Section 7.10.12).

   PFGEN_PFRSTAT.*PFRD* indication is cleared when GL_XLR_MARKER_TRIG_VMLR is written.
3. When hardware drain flow ends, hardware sets the PFGEN_PFRSTAT.*PFRD* and issues an interrupt to EMP.
4. Upon interrupt (or firmware polling of *PFRD*), EMP sees that PFGEN_PFRSTAT.*PFRD* is set, then does the following "post hardware drain" clean-ups
   a. Firmware releases all resource of the PF and its VFs (for example, nodes in PSM and in PE scheduler, function profile, switch population).
5. Firmware resets (zeros all fields) the QINT_TXQCTL, QINT_RXQCTL, and GLINT_CEQCTL registers for the queues of the PF.
   a. Disable the receive queues of the PF using the fast queue disable flow per each queue.
   b. EMP polls that *Transactions Pending* flag in the PCI configuration space of the PF and of all its VFs until they are cleared.
6. After all the previous steps are complete, firmware clears the *PFSWR* bit in the PFGEN_CTRL register. As a response to *PFSWR* clearing, hardware releases bus master cycles for the PF and its VFs.
   - Before clearing the *PFSWR* bit in the PFGEN_CTRL, the firmware must clear *VFSWR* bit in the respective VPGEN_VFRTRIG register for all the PF's VF, and the *VMSWR* bit in the VSIGEN_VFRTRIG register for all the PF's VSIs.
   - Firmware should clear the *VFLRS* bit of the corresponding VFs in the GLGEN_VFLRSTAT register.

The PF software polls the *PFSWR* bit until it is cleared (indicating that hardware completed its reset flow). Additionally, software should also poll the *Transactions Pending* flag in the PCI configuration space of the PF (indicating that all outstanding requests of the PF were completed).

- Once the *Transactions Pending* flag is cleared, the PF software can release the pinned memory structures.
- Once the *PFSWR* bit is cleared as well, the PF software can proceeds to the following steps.

  Note: If the *PFSWR* bit is not cleared within ~100 ms, it can be assumed that the shared data path inhibits the PFR completion. In such a case, the software can initiate the CORER to overcome any possible lock.

As part of the initialization flow, the PF driver checks the pending transactions of its VFs (by setting the VF index and the offset of the VF Device Status register in the PF_PCI_CIAA register, and then polling the pending flag in the PF_PCI_CIAD register). As part of the software/hardware initialization flow, the PF software should check for potential race condition with another PF or EMP initiating a global reset (CORER, GLOBR. or EMPR):

1. Query the reset counters (*CORERCNT*, *GLOBRCNT*, and *EMPRCNT*) in the GLGEN_RSTAT register.
2. Interrupt enable flow.
3. Read the GLGEN_RSTAT register for the device state (*DEVSTATE*) and the reset counters (*CORERCNT*, *GLOBRCNT*, and *EMPRCNT*).
   - If *DEVSTATE* = "Reset requested" or "Reset in progress", go to the global reset flow.
   - If *DEVSTATE* = "Device active", if the updated values of the global reset counters equal to the value sampled at the beginning of this procedure, then "all good". Software can continue with the rest of the software/hardware initialization flow.
   - Else, go to the global reset flow.

The PF should also read the PFINT_OICR register in case there were "other cause" interrupts during reset.

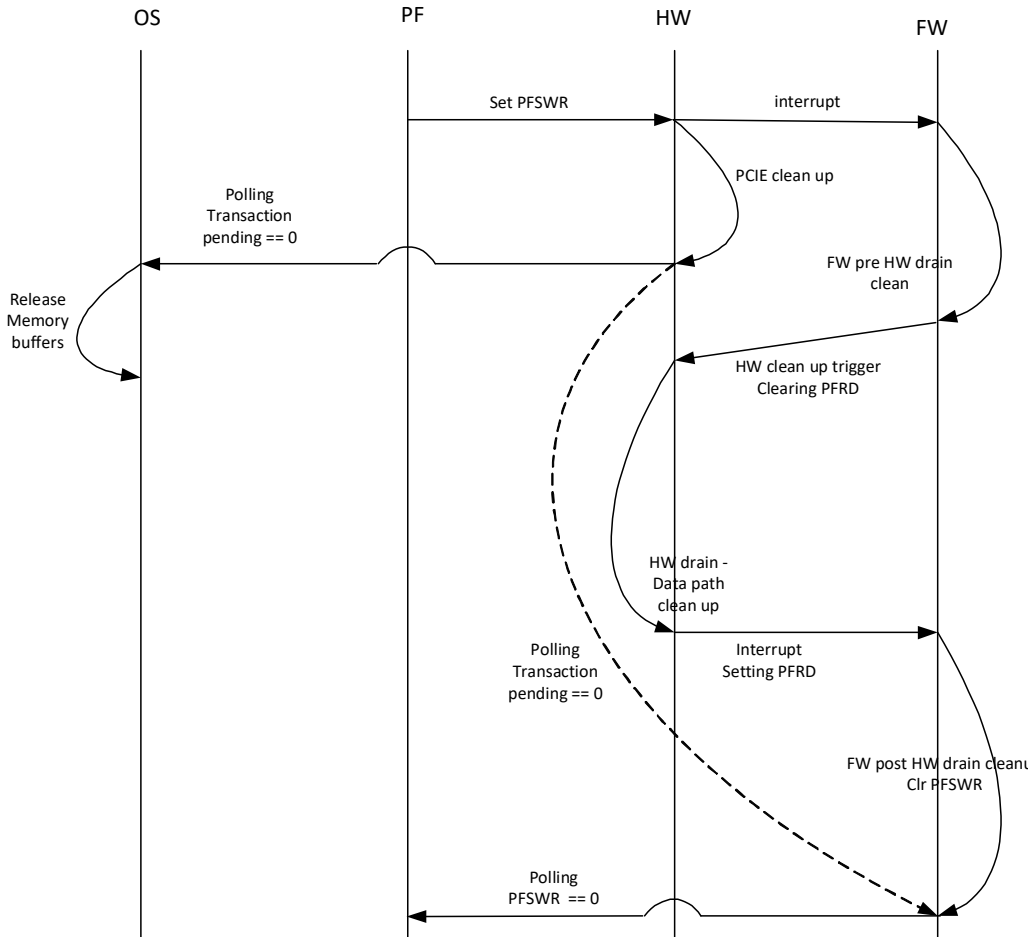


Figure 4-7. PFR Flow

4.1.3.2 FLR Flow

FLR resets a specific PF. In the case of SR-IOV, it also resets the VFs of this PF.

1. The following steps are optional:
   a. The operating system is expected to clear the *BME* bit in the Command register in the PCI configuration register of the PF.
   b. For SR-IOV, the operating system is expected to also clear the *BME* bit in the VF Command register in the PCI configuration register of all VFs of this PF.
   c. As a response, hardware avoids any new master cycles of the PF and its VFs (including MSI-X initiation).
   d. The operating system should poll the *Transaction Pending* bit in the Device Status register of the PF until it is cleared.
   e. For SR-IOV, the operating system should poll also the *Transaction Pending* bit in the VF Device Status register of all VFs of the PF until they are cleared.

   Note: All the previous steps are expected and recommended, but not enforced by the PCI specification.
2. The operating system sets the *FLR* bit in the Device Control register of the PF. The operating system is required by PCIe specification to wait 100 ms before it can assume that the FLR sequence is completed by hardware.
3. The PF hardware response as follows:
   a. Initialize the PCIe configuration space of the PF including clearing the *BME* bit of the PF and the *VFE* bit.
      - By clearing the *BME* bit, hardware avoids any further master accesses on the PCIe bus and discards any completions for the PF.
      - By clearing the *VFE* bit, all VFs of the PF avoid any further master accesses on the PCIe bus, discards any completions targeted for these VFs, and become hidden on the PCIe bus.
      - As part of the PCIe configuration initialization, the completion timeout is set to its hardware default.
      - Once all pending requests of the PF and its VFs are completed or completion timeout expires (whichever comes first), the T*ransaction Pending* bits in the PCIe configuration space are cleared.
   b. Auto-set the *PFSWR* bit in the PFGEN_CTRL register (triggering a PFR described in the section that follows).

Once the *Transaction Pending* bit in the PCIe configuration space of each VF is cleared, the operating system can release all VF memory structures and unload the VF driver (if required). Once the *Transaction Pending* bit in the PCIe configuration space of the PF is cleared, the operating system can release all PF memory structures and unload the PF driver (if required).

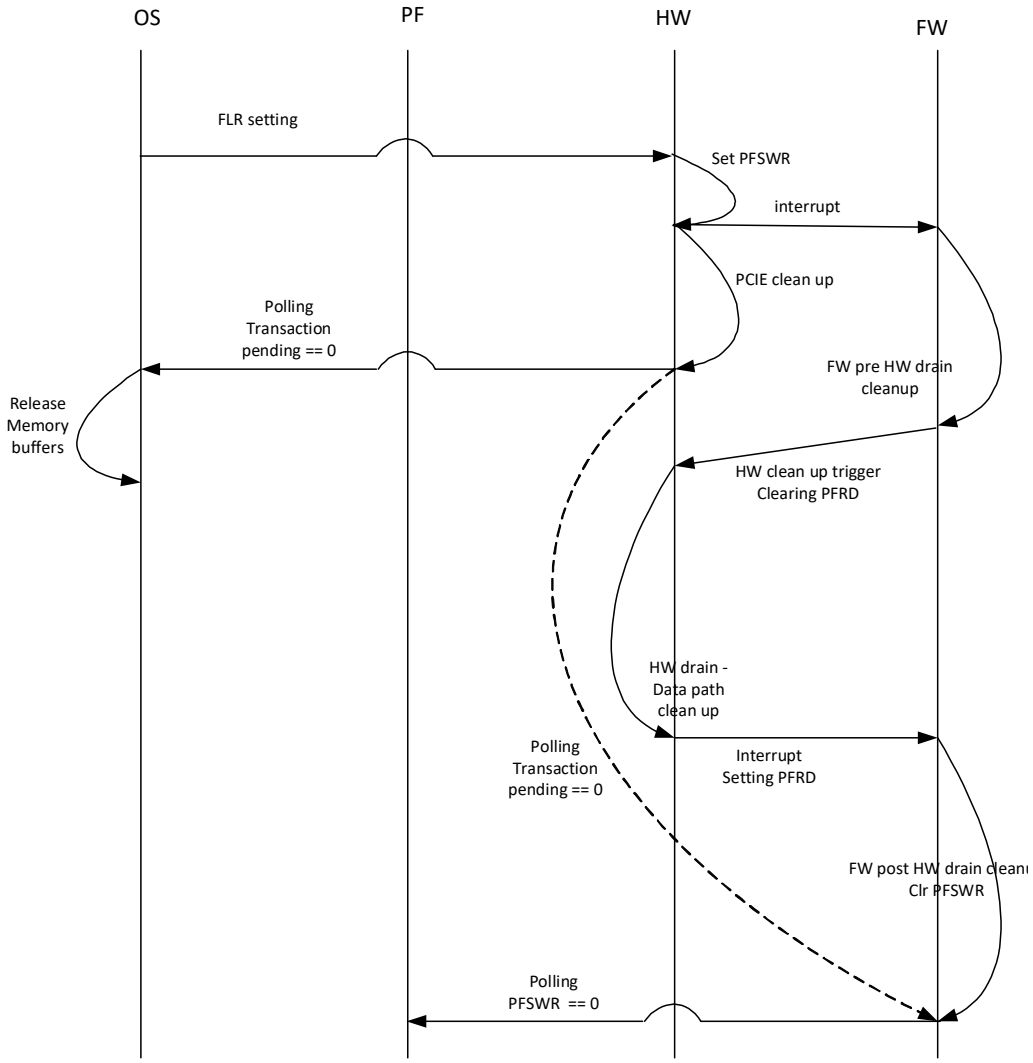


Figure 4-8. FLR Flow

4.1.3.3 VFR/VFLR Flows

The VF reset flow is shown in Figure 4-9 and detailed in the sections that follow.

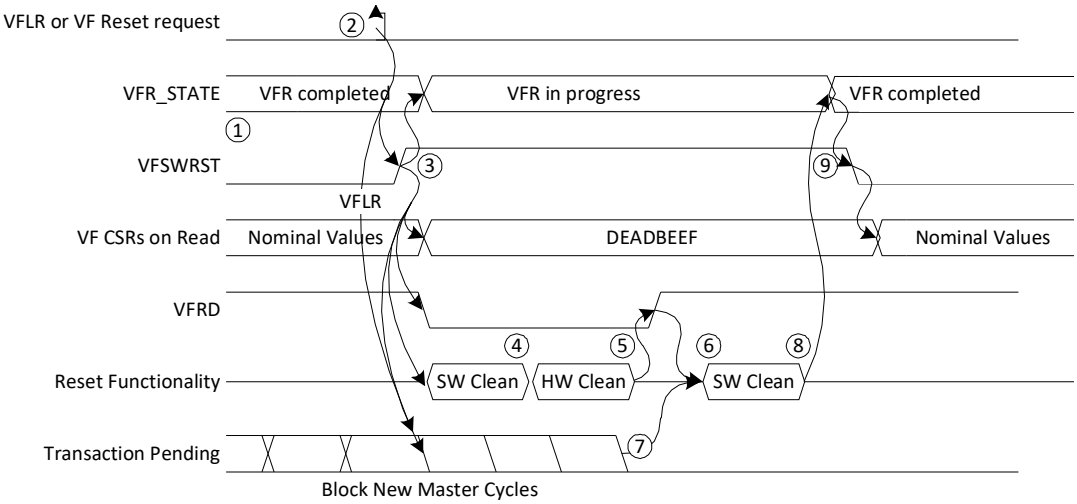


Figure 4-9. VF Reset (VFR) Flow

4.1.3.3.1 VF Reset Request by the VF Driver

The VF driver requests the VF reset from its parent PF as follows, and is shown in Figure 4-9. Also see Section 4.1.3.3.3, which describes the PF response to the VF reset request.

The *VFR_STATE* in the VFGEN_RSTAT register in this step is expected to be VFR completed.

Note: The VF software gets a control over its VF only after any prior VF reset flow is completed (step 1 in Figure 4-9).

1. The VF driver initiates a request to its parent PF to initiate a VF reset. The mechanism for sending this request is outside the scope of this document. It could be done using a VF-to-PF or any other software-based sideband channel (Step 2 in Figure 4-9).
2. The VF polls the *VFR_STATE* in the VFGEN_RSTAT register until it is set by the PF to VFR completed (Step 9 in Figure 4-9, and explained in Section 4.1.3.3.3).
3. The VF software proceeds activating the function.

At some point prior to requesting the PF for reset once again, VF must set *VFR_STATE* value to "reset in progress". When called, the PF software is called to initiate the VFR by the VF software. It sets the *VFSWR* bit in the VPGEN_VFRTRIG register of the VF (Step 3 in Figure 4-9) to initiate VFR (described in Section 4.1.3.3.3).

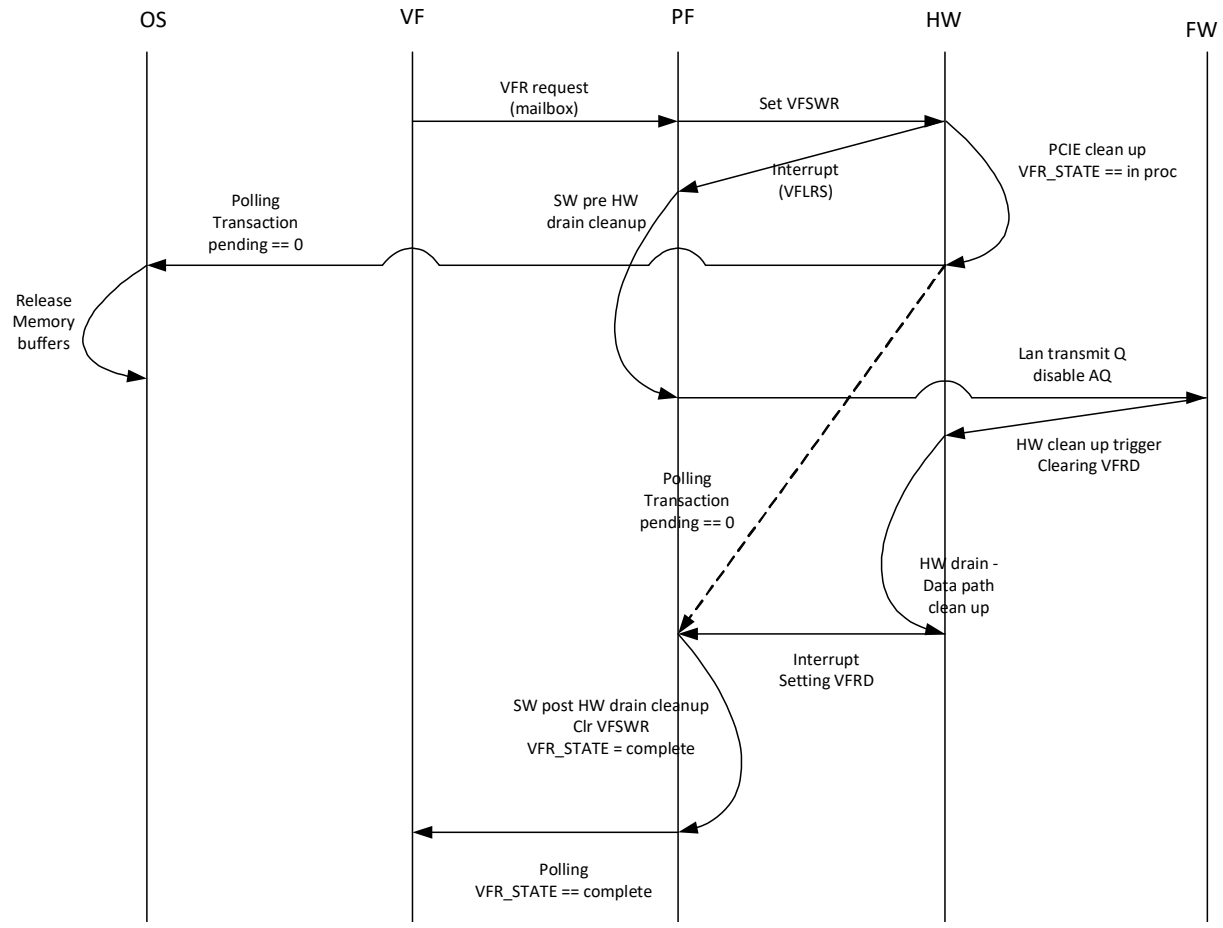


Figure 4-10. VFR Requested by VF Driver Flow

4.1.3.3.2 VF Reset Request by the Operating System (VFLR)

The VF reset initiation by the operating system is described as follows and shown in Figure 4-9.

1. The following steps are optional:
   a. Clear the *BME* bit in the VF Command register.
   b. As a response, hardware avoids any new master cycles of the VF (including MSI-X initiation). Old completions are trashed by hardware.
   c. The operating system polls the *Transaction Pending* bit in the VF Device Status register until it is cleared.
2. The operating system sets the *FLR* bit in the VF Device Control register (Step 2 in Figure 4-9).
3. The operating system is required by PCIe specification to wait 100 ms before it can assume that the VFLR sequence is completed by hardware.
4. If required, the operating system brings up a new VF (or the same VF). The VF software driver should poll the *VFR_STATE* in the VFGEN_RSTAT register until it equals to VFR completed (set by the PF software).

5. The VF software proceeds activating the function.

Hardware responses to VFLR are as follows:

1. Initialize the PCIe configuration space of the VF including the *Bus Master Enable* flag.
2. Set internally the *VFSWR* bit in the VPGEN_VFRTRIG register of the VF, which initiates a VFR (Step 3 in Figure 4-9, and explained in Section 4.1.3.3.3).

4.1.3.3.3 VF Reset Flow by the PF Software Driver

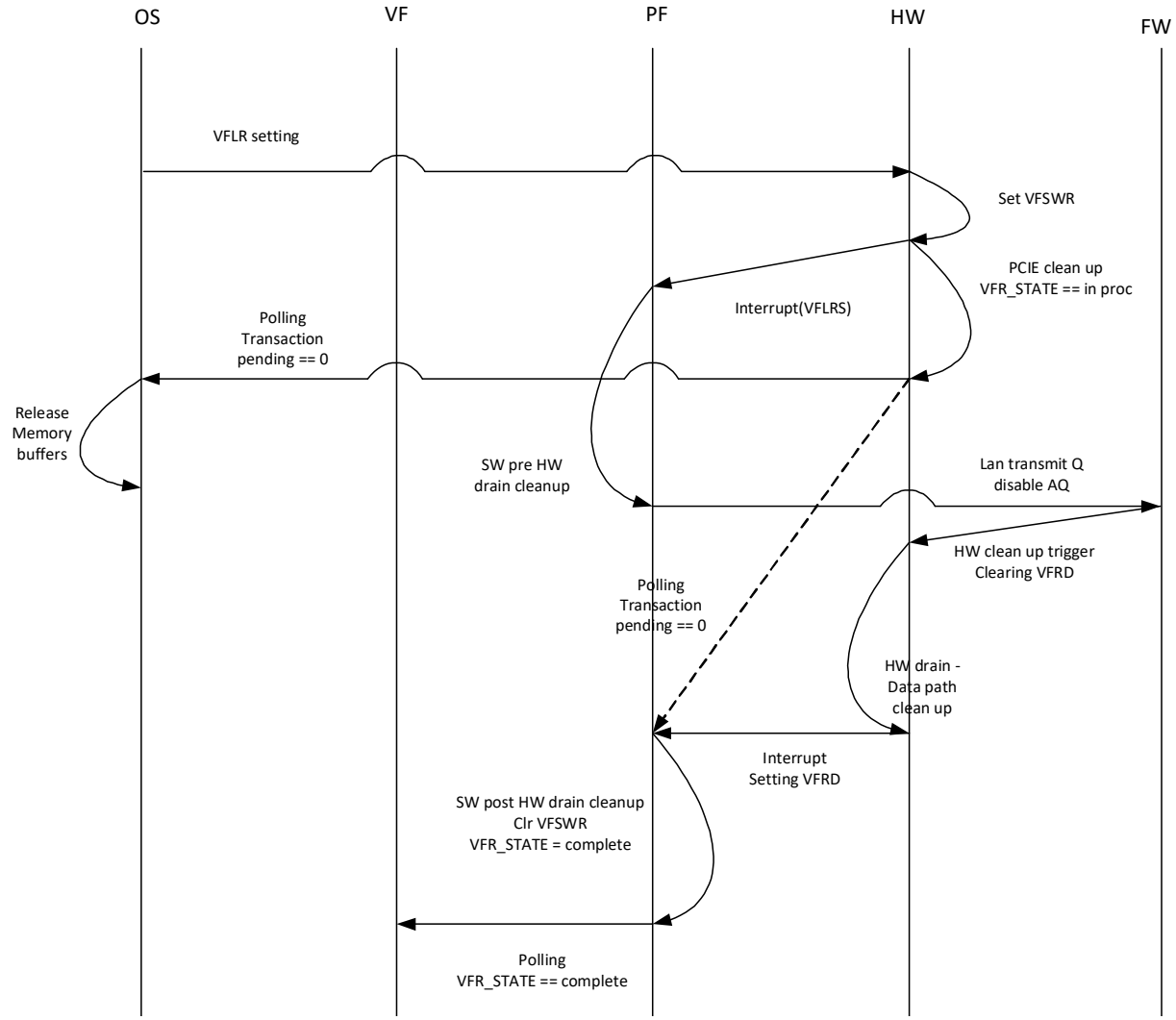


Figure 4-11. VFR Requested by OS

1. As a response to setting the *VFSWR*, hardware does the following:
   a. Reflects the *VFSWR* bit through *VFLRS* flags in the GLGEN_VFLRSTAT registers.
   b. Only in case of VFLR (OS initiated reset) an interrupt is issued to the parent PF. The PF response to the VFLR interrupt is described below.
   c. The hardware response to setting the *VFSWR* bit is listed in Table 4-1.

      Note: As part of the VF registers, the VFGEN_RSTAT register is cleared as well, reflecting VFR in progress state. Furthermore, hardware also gates read accesses to the CSRs of the VF returning DEADBEEF or DEADBEAF values and also gates any master accesses.

   d. The T*ransaction Pending* bit is cleared when there are no more pending completions. Once the *Transaction Pending* bit in the VF Device Status register is cleared, the operating system can release all VF memory structures and unload the VF driver (if required).
2. The PF software responds to the interrupt and queries the *VFLRS* flags in the GLGEN_VFLRSTAT registers to identify to which of its VFs a reset was issued (of the VFs that it owns). When set, PF software executes "pre hardware drain" clean-ups (Step 4 in Figure 4-9):
   a. PF software cleans up switch population of the VF.
   b. PF software cleans up ACL population of the VF (as described in Section 7.9.2.10).
   c. PF software cleans up RSS population of the VF (as described in Section 7.10.11.2).
   d. PF software cleans and disables FD and QH filters of the VF (as described in Section 7.10.12.1 and Section 7.10.12.2, respectively).
   e. PF software clears the enable bit of doorbell queues that are associated with the doorbell queues of the reset VF. Tx-Queues and Admin queues are automatically disabled by hardware
   f. When working in SR-IOV mode, PF software disables mailbox queues and sideband queues of the reset VF (as described in Section 9.5.3).
3. The PF software, via VM/VF Reset admin command, indicates firmware to initiate VF hardware drain (data path clean up) flow, it passes to firmware the Tx-Queue IDs of the VF (VM/VF Reset Admin queue command is described in Section 9.5.13.9). Queue IDs are described in the PF space. PF software must limit the amount of reset flows concurrently processed by hardware to maximum 4 for all its VFs and VMs (from GL_XLR_MARKER_TRIG* setting to data path clean in indication setting by hardware)
   a. EMP initiates and monitors the process of halting the Tx-Queues (as described in Section 9.5.13.9) via GL_XLR_MARKER_TRIG* registers, EMP indicates hardware to send a drain marker that drains the Tx data path, Rx data path, PE data path and logic orthogonally.
      - The specific GL_XLR_MARKER_TRIG* registers are:
        — GL_XLR_MARKER_TRIG_VMLR
        — GL_XLR_MARKER_TRIG_TCVMLR
        — GL_XLR_MARKER_TRIG_RCU_PRS
        — GL_XLR_MARKER_TRIG_PE (should be written only if PW is enabled)
   b. Draining the Tx and Rx data paths might take some time when the TC is paused by flow control. Hardware includes a timeout mechanism that prevents indefinite latency as described in Section 8.2.4.4.1.
   c. VPGEN_VFRSTAT.*VFRD* indication is cleared when GL_XLR_MARKER_TRIG_VMLR is written.

4. When hardware drain flow ends (reset flow is completed - Step 5 in Figure 4-9), hardware sets VPGEN_VFRSTAT.*VFRD* and issues an interrupt to parent PF.
5. Once the *VFRD* flag in the VPGEN_VFRSTAT register is found active, the PF software writes 1b to clear to the matched *FVLRS* bit in the GLGEN_VFLRSTAT registers for the VF under reset.

   Note: The GLGEN_VFLRSTAT registers are composed of 256 bits for the 256 VFs. All PFs have access to the bits of all VFs. However, it is expected that the PFs writes 1b to clear to those bits that match its VFs and only to those bits.)

6. After that PF software proceeds with the VF reset flow (Step 6 in Figure 4-9) and executes the "post hardware drain" clean-ups:
   a. Disable the receive queues of the VF following the fast queue disable flow per each queue as described in Section 10.4.3.1.2.

      Note: The hardware auto-disables the transmit queues of the VF.

   b. PF releases all resource of the VF (for example, nodes in PSM and in PE scheduler, function profile, switch population, and queue allocation as described in Section 10.2.1 and Section 10.2.2).
   c. PF resets (zeros all fields) the QINT_TXQCTL, QINT_RXQCTL, and GLINT_CEQCTL registers for the queues of the VF.
   d. Resets the VPINT_ALLOC, VPINT_ALLOC_PCI, and GLINT_VECT2FUNC registers of the VF (registers that define the vectors assignment to the VF).
7. The PF driver checks the pending transactions of its VF (by setting the VF index and the offset of the VF Device Status register in the PF_PCI_CIAA register and then polling the pending flag in the PF_PCI_CIAD register) (Step 7 in Figure 4-9).
8. The following steps in this bullet item are part of re-enabling the VF. It can be executed at this phase before notifying the VF that the reset flow is completed or at a later phase (depending on software implementation):
   a. Add the VSIs for the VF (including its Transmit and Receive queues and its scheduler).
   b. Add the MAC/VLAN filters for the VSI.
   c. Enable the VFLAN_QTABLE by setting the VPLAN_MAPENA and then program the VFLAN_QTABLE to the queues of the VF.
9. The PF software completes the flow by notifying the VF that the reset flow is completed. It sets the *VFR_STATE* field in the VFGEN_RSTAT register to VFR completed and clears the *VFSWR* bit in the VPGEN_VFRTRIG register (Step 8 in Figure 4-9).
10. Hardware response to cleared VFSWR flag:
    a. The VF CSRs are unlocked. As a result, the VF software can see the updated *VFR_STATE* in the VFGEN_RSTAT register that equals to VFR completed (Step 9 in Figure 4-9).
    b. Master cycles are not blocked anymore by the reset logic.
    c. The hardware is ready to be used by the VF.

4.1.3.4 VMR Flow

1. PF sets the *VMSWR* bit in the VSIGEN_RTRIG register of the VM.
2. PF initiates "pre hardware drain" clean-up flow.
   a. PF software cleans up switch population of the VM.
   b. PF software cleans up ACL population of the VM (as described in Section 7.9.2.10).
   c. PF software cleans up RSS population of the VM (as described in Section 7.10.11.2).
   d. PF software removes the QH and FD filters of the VM.
   e. PF software sends dummy doorbell for all the doorbell queues of the VM and waits for all dummy doorbells (generated at "pre hardware drain" flow) to be pulled from doorbell queues. Dummy doorbell is a regular doorbell description that has its *RS* and *Dummy* fields set to 1 (doorbell descriptor format is described at Section 10.5.5.6).
   f. When working in PASID mode, PF software disables mailbox queues of the reset VM.
   g. PF software removes the interrupt causes of the VM from the active linked list (as described in Section 9.1.3.1.2).
3. As a response to VMSWR bit setting, hardware does the following:
   a. The hardware response to setting the *VFSWR* bit is listed in Table 4-1.

      Note: As part of the VM registers, the VSIGEN_RSTAT register is cleared.

   b. The *Transaction Pending* bit is cleared when there are no more pending completions. Once the *Transaction Pending* bit in the VM Device Status register is cleared, the operating system can release all VM memory structures.
4. The PF software, via VM/VF Reset admin command, indicates firmware to initiate VM reset flow. It passes to firmware the VM Tx-Queues identifiers (VM/VF Reset Admin Queue command is described in Section 9.5.13.9). Queue IDs are relative to the PF space.
5. PF software must limit the amount of reset flows concurrently processed by hardware to maximum 4 for all its VFs and VMs (from GL_XLR_MARKER_TRIG* setting to data path clean in indication setting by hardware).
   a. EMP initiates and monitor the process of halting the Tx-Queues (as described in Section 9.5.13.9) via GL_XLR_MARKER_TRIG* registers, EMP indicates hardware to send a drain marker that drains the Tx data path, Rx data path, and PE data path and logic orthogonally.
      - The specific GL_XLR_MARKER_TRIG* registers are:
        — GL_XLR_MARKER_TRIG_VMLR
        — GL_XLR_MARKER_TRIG_TCVMLR
        — GL_XLR_MARKER_TRIG_RCU_PRS
        — GL_XLR_MARKER_TRIG_PE (should be written only if PW is enabled)
      - Draining the Tx and Rx data paths might take some time when the TC is paused by flow control. Hardware includes a timeout mechanism that prevents indefinite latency as described in Section 8.2.4.4.1.
      - VSIGEN_VFRSTAT.*VMRD* indication is cleared when GL_XLR_MARKER_TRIG_VMLR is written.
   b. When hardware drain flow ends (reset flow is completed - Step 5 in Figure 4-9), Figure 4-9 sets the VPGEN_VFRSTAT.*VFRD* and issues an interrupt to parent PF.

6. After that, PF software proceeds with the VF reset flow (step 6 in Figure 4-9) and executes the "post Figure 4-9 drain" clean-ups.
   a. Disables the receive queues of the VF following the fast queue disable flow per each queue.
   b. Sends a CQ drain marker for all the completion queue of the VMs. it waits for all completions to be pulled from completion queues. CQ drain marker is generated by writing GLCOMM_CQ_CTL with the CMD field set to *VM_reset_marker* value.
   c. PF releases all resource of the VM (For example, nodes in PSM and in PE scheduler, function profile, switch population). The PF driver checks the pending transactions of its VM and waits until it is cleared.
   d. PF resets (zeros all fields) in the QINT_TXQCTL, QINT_RXQCTL, and GLINT_CEQCTL registers for the queues of the VM.
7. The PF driver checks the pending transactions of its VM and waits until it is cleared.
8. The PF software clears the *VMSWR* bit in the VSIGEN_VFRTRIG register.
9. Hardware response to cleared *VMSWR* flag:
   a. Master cycles are not blocked anymore by the reset logic.
   b. The hardware is ready to be used by the VM.
10. After PF driver polls the VMR done indication in the VSIGEN_RSTAT register, it:
    a. Polls the *Transactions Pending* flag of the VM, verifying that there are no transaction pending of the VM as follows:
       - Set the VSI index in the PFPCI_VMINDEX and then poll the PFPCI_VMPEND register.
    b. Does not clear the interrupts settings that might be shared with the PF (or other VMs).
    c. Does not check pending transactions of the VM that does not exist.
    d. Completes the flow by clearing the VMSWR bit in the VSIGEN_RTRIG register.

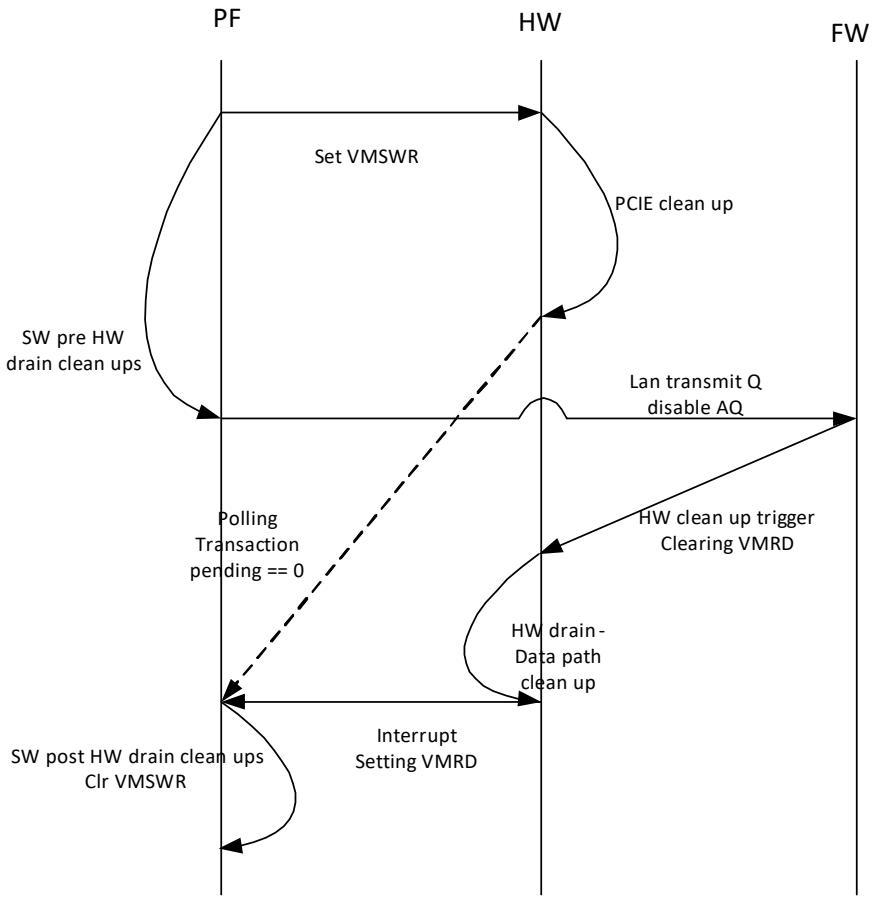


Figure 4-12. VMR Flow

Notes: Overall number of reset flows concurrently processed by hardware is limited to 40 (from GL_XLR_MARKER_TRIG* setting to data path clean in indication setting by hardware):

- Max 32 VF or VM reset flows - PF software limits the amount of reset flows concurrently processed by hardware to maximum 4 for all its VFs and VMs.
- Max 8 PF reset flows for 8 PFs.

4.2 Power-On and Reset

This section describes the flow of the E810 power-up. The initialization sequence for the E810 is broken down into phases: power on, BIOS initialization, and software device driver load.

- Power on (Section 4.1.2.1) is the first phase that includes all steps required to support pre-boot power management and manageability, satisfy the PCIe requirements for exiting cold reset, and prepare for the BIOS initialization phase. This stage also covers firmware initialization.
- BIOS initialization (Section 4.3) begins when Option ROM code is loaded by BIOS to provide boot services for PXE. Other Option ROM capabilities include configuration images for certain OEM environments, such as SMASH/CLP.
- The final phase of the E810 initialization is the software device driver load, where the operating system loads the various software device drivers and the E810 has been initialized for post-boot operation. See Section 4.4.

4.2.1 Auto-Load Shadow RAM

This phase of auto-load 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 3.4.4.2 for more details on how this is done.
- If the attached Flash device is properly configured, auto-load 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.

4.2.1.1 Auto-Load 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 in Table 4-2 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 ready to use.

Default hardware values:

• GLNVM_ULD.*_*DONE* = 0b

The flow during a power-on stage is as follows:

- If the NVM is found blank:
  - All GLNVM_ULD.*_*DONE* bits are set by the E810.
- If the NVM is found valid:
  - After process is done by EMP firmware, it sets the GLNVM_ULD.*_*DONE* bit to 1b.

4.2.1.2 Firmware Initialization

There are two sets of firmware loaded for the E810. The first is for the Embedded Management Processor (EMP). This firmware is first to load and is required for all E810 deployments. It is optionally followed by loading Protocol Engine firmware required for RDMA and UDA operation.

Once EMP firmware loads, the following E810 features are enabled:

- The external Ethernet link is active.
- Default internal switching components have been configured.
- Communication with the BMC is possible if supported by the E810 and the system (optional).
- OEM-specific manageability agents are active and responding to commands from the Ethernet fabric (optional).
- Admin queues for all 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 E810 for device driver initialization (see Section 9.5.3).
- HMC default profile has been configured.

Once EMP firmware is up and running, the E810 can be configured via its management interfaces, and certain device capabilities are then enabled or disabled (such as soft SKUing).

Protocol Engine firmware is responsible for processing configuration requests for RDMA and UDA functionality that is provided by the Protocol Engine. See Section 11.5.2 for more information on the specifics of the commands supported by Protocol Engine Firmware. Protocol Engine firmware load begins once the EMP firmware has finished loading and initialization and only after primary reset was de-asserted. When the Protocol Engine firmware is loaded and initialized, the GLPE_CPUSTATUS0-2 registers indicate that it is safe to start using Protocol Engine capabilities.

Protocol Engine firmware is loaded only when the protocol engine is enabled in the NVM.

4.2.1.3 MAC Address Initialization

The E810 supports a port MAC Address for each of its ports. These addresses are used by the internal switch for L2 filtering of packets, by the classification filters, for WoL purposes, and for link-level functionality, such as flow control frames and LLDP. The addresses are loaded from the NVM PRTPM_SAL and PRTPM_SAH registers.

Table 4-4 lists how each address can be set or read and where it is stored.

Table 4-4. MAC Address Information

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: • Internal switch entries • PRTPM_SAL/H • PRTMAC_HSEC_CTL_TX_SA_PART1/2		N/A	Yes
LAN MAC Address – BIOS setting	Alternate RAM (Current LAN MAC Address (Low/High)) Loaded by firmware to: • Internal switch entries • PRTPM_SAL/H • PRTMAC_HSEC_CTL_TX_SA_PART1/2	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	PRTPM_SAL/H	Manage MAC Address read (WoL address valid)	Manage MAC Address Write AQ command (update LAA and WoL address).	Yes
WoL MAC Address preserve on PFR	RAM	Manage MAC Address read (WoL_preserve_on_PFR)	Manage MAC Address Write AQ command (WoL_preserve_on_PFR). Set by software before D0->Dr/D3 transition. Cleared by software after returning to D0). 0 by default after POR/ EMPR.	No
LAA MAC Address	Local firmware variable	Manage MAC Address read (LAN address valid)	AQ command (update LAA address) ***Note:*** Changing the LAA only does not impact the address used to detect Magic Packets.	No
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

The following rules apply:

- The *PF_NUM* field in the PRTPM_SAH register is deducted from the PF issuing the command.
- The *AV* field in PRTPM_SAH 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.

4.2.1.3.1 Manage MAC Address Read Command (0x0107)

This command is used by the PF driver to read the per-PF station MAC Address. This is an indirect command.

Table 4-5. Manage MAC Address Read Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0107	Command opcode.
Datalen	4-5	0x0	0x18 (the response buffer size).
Return Value/VFID	6-7	0x0	Return value. Zeroed by device driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Command Flags	16-17	Reserved	Reserved.
Port Number	18-19.0	Port Number	**Byte 18:** Logical Port Number This field specifies the logical port number, and is used when the physical function owns more than one port. A value of 0xFF means that a dump of all ports addresses is requested. A port number of 20 (logical port number of the E810) returns the local MAC Address of the E810. **Bit 19.0:** Logical Port Number is valid.
	19.1-19.7	Reserved	Reserved.
Reserved	20-23	0x0	Reserved.
Data Address High	24-27	Buffer Address	High bits of the return buffer address.
Data Address Low	28-31	Buffer Address	Low bits of the return buffer address.

4.2.1.3.1.1 Manage MAC Address Read Response (0x0107)

A firmware acknowledge to the Manage MAC Address Read command. This is an indirect response.

Table 4-6. Manage MAC Address Read Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0107	Command opcode.
Datalen	4-5	0x0	0x18 (the response buffer size).
Return Value/VFID	6-7	Return Value	Return value. Firmware supplies in the *Return Value* field an indication on the completion of the Manage MAC Address Read command. 0x0 = No error. Others = Error detected in the command.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Command Flags	16.0-16.3	Reserved	Zeroed by the EMP.
	16.4	LAN Address Valid	This bit is set if any of the addresses reported in the buffer is of type LAN.
	16.5	SAN Address Valid	Reserved for the E810.
	16.6	Port Address Valid	Reserved for the E810.
	16.7	WoL Address Valid	This bit is set if any of the addresses reported in the buffer is of type WoL.
	17.0	MC_MAG_EN	If set, multicast Magic Packets generate a WoL event (if WoL is enabled for this function).
	17.1	WoL_preserve_on_PFR	Set to preserve WoL MAC on PFR (to be set by software before D0->Dr/D3 transition. Cleared by software after returning to D0 state). 0 by default after POR/EMPR. Relevant only if WOL MAC was set.
	17.2-17.7	Reserved	Reserved.
Reserved	18-23	0x0	Reserved.
Data Address High	24-27	Buffer Address	High bits of the return buffer address.
Data Address Low	28-31	Buffer Address	Low bits of the return buffer address.

Note: All MAC Addresses are a byte stream in network order.

The buffer is as follows:

Address	Offset	Description
Reserved	0-1	Reserved.
PF LAN SA	2-7	Current device value of the PF LAN MAC Address. Validated by *LAN Address Valid* flag. This address is returned from LAA address if valid, otherwise from Alternate RAM if valid, otherwise from the NVM if valid.

4.2.1.3.2 Manage MAC Addresses Write Command (0x0108)

This command is used by the PF driver to write the per-PF station MAC Address. This is a direct command.

Table 4-7. Manage MAC Address Write Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0108	Command opcode.
Datalen	4-5	0x0	Must be 0x0. Value is ignored.
Return Value/VFID	6-7	0x0	Return value. Zeroed by device driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Command Flags	16	Port Number	Port Number This field specifies the port number, and it is used when the physical function owns more than one port.
	17.0	MC_MAG_EN	Used to set the *MC_MAG_EN* bit.
	17.1	LAA_WOL_PRESERVE	LAA WoL preserve on PFR Relevant only if WOL MAC was set.
	17.2	Valid	Port number in byte 16 is valid.
	17.5-17.3	0x0	Reserved.
	17.7-17.6	Write Type	00b = Update LAA only. 01b = Update LAA and WOL address. All other values are reserved.
SAH	18-23	MAC Address	MAC Address as a byte stream in network order.
Reserved	24-31	Reserved	Reserved.

4.2.1.3.2.1 Manage MAC Address Write Response (0x0108)

A firmware acknowledge to the Manage MAC Address Write command. This is a direct response.

Table 4-8. Manage MAC Address Write Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0108	Command opcode.
Datalen	4-5	0x0	Must be 0x0. Value is ignored.
Return Value/VFID	6-7	Return value	Return value. Firmware supplies in the *Return Value* field an indication on the completion of the Manage MAC Address Write command. 0x0 = No error. Others = Error detected in the command.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Command Flags	16	Port Number	Port Number This field specifies the port number, and it is used when the physical function owns more than one port. Copied from command.
	17.0	MC_MAG_EN	Used to set the *MC_MAG_EN* bit.
	17.1	LAA_WOL_PRESERVE	LAA WoL preserve on PFR Relevant only if WOL MAC was set.
	17.2	Valid	Port number in byte 16 is valid.
	17.5-17.3	0x0	Reserved.
	17.7:17.6	Write Type	00b = Update LAA only. 01b = Update LAA and WOL address. All other values are reserved.
Reserved	18-31	Reserved	Reserved.

4.2.1.4 Power-On Device State

This section describes the specific setting of each of the E810's components required for pre-boot operation. It describes the information loaded from the NVM (per component) and the state attained by each.

- **Manageability** — System management functionality, if enabled, is fully operational by the end of the EMP firmware initialization sub-stage. Some configuration is loaded from the NVM during the power-on stage. Capabilities might include the following:
  - — Sideband interfaces
  - — Pass-through manageability (including packet filtering)
  - — Preparation for OS-to-BMC traffic
  - — Preparation for MCTP over PCIe operation

- **Internal MAC and PHY** — If either system management or APM WoL are enabled, enabled LAN ports are brought up by firmware once EMP firmware initialization completes. Otherwise, enabled LAN ports are brought up following PERST# de-assertion.
  - See Section 3.2 for more details.
- **Admin Queue (AQ)** — A queue (Tx and Rx pair) is activated per enabled PCI function. For example, an AQ is needed for BIOS to change the switch or scheduler configuration.
  - See Section 9.5.3 for more details.
- **Internal Switch** — The internal switch is configured from the NVM for basic switching capabilities to the EMP and the enabled PCI functions. First, the switch programmable logic is loaded, followed by configuration of a basic switch topology. The topology is for basic L2 functionality.
- **DCB** — The NVM loads the LLDP and DCBX setting into the E810. Once EMP firmware initializes and link is up, firmware engages in DCBX negotiation. Firmware then makes the appropriate changes in the E810's configuration based on the outcome of the DCBX protocol. At this stage, DCB capabilities (TCs, ETS, PFC) can be enabled.
- **Tx-Scheduler** — As the switch VSIs are enabled, firmware allocates Tx-Scheduler per each PF VSI based on the NVM configuration. Each queue is configured to default behavior. Firmware also generates the required handles to enable system software to update the configuration of each queue. If DCBX protocol runs, the outcome of the protocol exchange might translate into changes in scheduler configuration.
- **Host Memory Cache (HMC)** — NVM settings supply the initial HMC configuration using a simple set of rules that enable equal distribution of HMC resources to all PFs that are connected to external Ethernet ports.
- **Power Management** — Some power management capabilities are supported pre-boot or during BIOS initialization.
  - The E810 might be enabled for APM wake during power up. See Section 5.3.1 for more details on how APM Wake is configured.
  - EEE might be enabled once EMP firmware is initialized.
- **LAN** — LAN queues are available for network boot in the BIOS initialization phase. Some LAN configuration is loaded from the NVM during power on, including partitioning of the LAN queues among PFs.
- **Protocol Engine (PE)** — NVM setting can disable PE operation in the device. The Protocol Engine is not enabled during Power-On and BIOS Initialization stages. It is configured and initialized in the Driver Load phase.

4.3 BIOS Initialization

This section describes how BIOS code might update the E810's configuration and the capabilities for network boot. The following sections are provided:

- Section 4.3.2 describes how BIOS code might change the E810's configuration on each boot.
- Section 4.3.3 describes some aspects of supporting network boot.
- Section 4.3.4 describes the specific setting of each of the E810's components required for the BIOS initialization stage

4.3.1 Initial State

The state of the E810 when BIOS begins to access it is described in Section 4.3.4.

BIOS code must check that the EMP completed device initialization. This is done through the Get Version AQ command described in Section 9.5.13.1.

4.3.2 Non-Persistent Configuration

During power up, the E810 is factory-configured from the NVM. However, the factory configuration can be overridden in two possible ways:

- **Persistent Configuration** — System tools (such as software agents and SMCLP commands) can 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 can be restored at a later time.
- **Non-Persistent Configuration** — An on-die Alternate RAM structure is provided to store configuration values that 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 4-9.

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).

4.3.2.1 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 (1.5K DWords), where address 0x000 points to the beginning of the memory.

During a cold reset sequence, the alternate structure can be written by either an external system management agent (via the device management interfaces) or by BIOS level code (like SMASH/CLP commands).

The structure is partitioned as follows:

Section	Address (DW)	Size (DW)	Content
Per PF	0-511	512-64 per PF	8 per-PF sections, one per PF. These sections are described in Section 4.3.2.1.1 and can be accessed by each PF.
Reserved	512-1023		Reserved for more functions.
EMP	1024-1536	512	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.

4.3.2.1.1 Per-PF Sections

Table 4-9. 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	2-18	Reserved	N/A	N/A	Never
PF	19	RDMA	Yes	Yes	POR
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, 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 is not applied, even if the condition is met. Issuing the "Done Alternate Write" command when it is not explicitly required does 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 must be written back to the Flash using the NVM Write Activate command (Section 3.4.10.8).

4.3.2.1.1.1 Current LAN MAC Address (Offset 0x0, 0x1)

Lower DWord:

Field	Bit(s)	Description
MAC Address	31:0	MAC Address Contains the LS 32-bit of the address.

Upper DWord:

Field	Bit(s)	Description
MAC Address	15:0	MAC Address Contains the MS 16-bit 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.

Upper DWord:

Actions taken on a change in this entry:

- When valid, it overrides the MAC Address loaded from the NVM.

4.3.2.1.1.2 RDMA (Offset 0x19)

Field	Bit(s)	Description
Override Enable	0	Enable BIOS override of netlist configuration recommendation for topology of 4 ports and below.
RDMA Enable	1	Enable RDMA for topology of 4 ports and below.
Reserved	30:2	Reserved.
Valid	31	Valid bit 0b = DW is invalid and should be skipped. 1b = Entry is valid and should be processed.

4.3.2.2 AQ Commands

Table 4-10 lists the different AQ commands used to manage the alternate structure.

Table 4-10. 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.	4.3.2.2.1
Write Alternate - Indirect	0x0901	Write a block of parameters into the alternate structure.	4.3.2.2.2
Read Alternate - Direct	0x0902	Read up to two parameters from the alternate structure.	4.3.2.2.3
Read Alternate - Indirect	0x0903	Read a block of parameters from the alternate structure.	4.3.2.2.4
Done Alternate Write	0x0904	Indication that all CLP strings (for the entire E810 in legacy BIOS mode, and per LAN Port in UEFI mode) have been sent to the E810.	4.3.2.2.5
Clear Port Alternate	0x0906	Clear content of Alternate RAM relevant to this port.	4.3.2.2.6

4.3.2.2.1 Write Alternate - Direct Command (0x0900)

The Write Alternate - Direct command writes to the alternate structure up to two parameters.

Table 4-11. Write Alternate - Direct Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the 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 4-12. Completion for the Write Alternate - Direct Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.2 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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-31		Reserved. Might contain the values sent in the original command.

4.3.2.2.2 Write Alternate - Indirect Command (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 4-13. Write Alternate - Indirect Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the 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 4-14. Completion for the Write Alternate - Indirect Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.2 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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-31		Reserved.

4.3.2.2.3 Read Alternate - Direct Command (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 4-15. Read Alternate - Direct Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the 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 4-16. Completion for the Read Alternate - Direct Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.2 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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the 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.

4.3.2.2.4 Read Alternate - Indirect Command (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 4-17. Read Alternate - Indirect Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the 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 4-18. Completion for the Read Alternate - Indirect Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.2 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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the 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.

4.3.2.2.5 Done Alternate Write Command (0x0904)

The Done Alternate Write command indicates to the E810 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 E810 (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 E810'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 3-62).

Table 4-19. Done Alternate Write Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the 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 4-20. Completion for the Done Alternate Write Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the 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.

4.3.2.2.6 Clear Port Alternate Write Command (0x0906)

The Clear Port Alternate command indicates to the E810 to clear the alternate sections of the PF. The port is inferred from the PF that sent the command.

Table 4-21. Clear Port Alternate Write Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the 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 4-22. Completion for the Clear Port Alternate Write Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-31	0x0	Reserved.

4.3.2.3 Example of a SMASH/CLP Flow - Legacy BIOS

The following pseudo code provides an example of how such a flow might be performed by legacy BIOS using SMASH/CLP commands. The following guidelines should be kept:

- If a certain PCI function is disabled via this mechanism, pre-boot software does not access any resource of that function (such as any CSR) once it sends the Done Alternate Write AQ command. The E810 confirms the command, resets, and disables the function.
- By the time the SMASH/CLP commands are executed and a function is disabled, there is no Tx/Rx activity in the E810 (since no queues have been initialized).
- All ports enabled in the NVM have the Option ROM enabled (such as the lowest PCI function per port has an Expansion ROM BAR).
- BIOS calls all CLP entry points for all functions before getting into the initialization phase.
- It is not guaranteed that a system reset is issued immediately following this sequence. For example, the configuration settings must take place even if such a reset is not issued.
- During the initialization phase, pre-boot software has to issue a global reset followed by a Start LLDP Agent AQ command to start the LLDP agent in firmware.

```
// Initial state:
// The device is configured from NVM.
// The following flow runs per each device.

BIOS does PCI enumeration and discovers functions with Option ROM enabled.

For each LAN Port with Option ROM enabled // sent to the lowest number PCI function on the port,
    BIOS loads the Option ROM into system RAM.
    Set First-Init to true // When Init is later called, it's the 1st call to Init.

    For each CLP string received by BIOS for a function within the port,

        If the Option ROM for the port supports SMCLP entry point,
            BIOS calls SMCLP entry point for port with CLP string.
            SMCLP section in Option ROM processes the CLP string.
            Option ROM keeps track of SMCLP status for the port.
        End-If

    End-For
End-For

For each LAN port with Option ROM enabled,
    BIOS calls INIT entry point for the port.
End-For

SMCLP Entry Point:

If CLP type is SET and action is “Return to default” // clear any pre-existing CLP configuration,
    Send the "Clear Port Alternate" admin command to invalidate all Alternate Memory parameters
    for the port and its PFs.
End-If

If CLP type is SET,
    Generate “Write Alternate - Direct” admin command(s) to the device.
End-If

If CLP type is EXIT // Apply any configuration changes from previous commands that have not
been applied,
    Send appropriate admin commands with default values for the PF parameters not specified by
    the CLP string that have a default behavior in this mode, different than the hardware
    default.
End-If

End SMPCLP Entry Point.

SMCLP INIT Entry point:

If First-Init is true // the flow below should only be executed on the 1st call to Init,

    If SMCLP status is false // means no CLP strings have been received for any functions for the
    device,
        // Option ROM INIT configures the device in a standard operating mode (i.e. not an OEM
        specific mode).
    End-If

    If SMCLP status is true // means at least one function had CLP strings,
        // This is the time to load the CLP parameters from the Alternate Structure into the
        device.
        Send "Done Alternate Write" command indicating end of CLP strings for the device.
    End-If

    // Global Reset should be done only once per device.

    If firmware required a reset in the "Done Alternate Write" response,
        Issue Global Reset of the device.
        Set First-Init to false.
    End-If

End-If
```

```
Continue with INIT code...

End SMCLP INIT Entry Point.
```

4.3.2.4 Example of a SMASH/CLP Flow - UEFI

The following pseudo code provides an example of how such a flow might be performed by the UEFI drivers using SMASH/CLP commands. The following guidelines should be kept:

- If a certain PCI function is disabled via this mechanism, pre-boot software does not access any resource of that function (such as any CSR) once it sends the Done Alternate Write AQ command for that function. The E810 confirm the command, resets, and disables the function.
- By the time the SMASH/CLP commands are executed and a function is disabled, there is no Tx/Rx activity in the E810 (since no queues have been initialized).
- A UEFI driver is executed for each enumerated LAN port. For example, any port enabled in the NVM and not disabled by strapping.
- It is not guaranteed that a system reset is issued immediately following this sequence. For example, the configuration settings must take place even if such a reset is not issued.

```
// Initial state:
// The device is configured from NVM.
// The following flow runs per each device.

BIOS does PCI enumeration and discovers PCI functions.

For each enabled LAN Port,
    BIOS calls driver START entry point.

    For each CLP string received by BIOS for a function within the port,
        SMCLP section processes the CLP string.
    End-For

    Driver sends a "Done Alternate Write" command indicating end of CLP strings for the port.

    If firmware required a reset in the "Done Alternate Write" response,
        Issue Global Reset of the device.
    End-If

End-For

SMCLP section in START:

If CLP type is SET and action is "Return to default" // clear any pre-existing CLP configuration,
    Send the "Clear Port Alternate" admin command to invalidate all Alternate Memory parameters
    for the port and its PFs.
End-If

If CLP type is SET.
    Generate "Write Alternate - Direct" admin command(s) to the device.
End-If

If CLP type is EXIT // Apply any configuration changes from previous commands that have not been
applied,
    Send appropriate admin commands with default values for the PF parameters not specified by
    the CLP strings that have a default behavior in this mode, different than the hardware
    default.
End-If

End SMPCLP section.
```

4.3.2.5 Processing the Alternate Structure

As a result of each Done Alternate Write AQ command, the EMP loads the relevant alternate structure parameters into the E810 according to the following sequence:

- EMP loads any required parameters from the alternate structure into the hardware.
- EMP ACKs the Done Alternate Write command.
- Global 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.

4.3.3 Network Boot

Any functions can be a PXE or UEFI boot function (up to 8 such functions).

4.3.4 Device State

This section is limited to changes in the E810's state made in the BIOS initialization stage. For units not mentioned here, behavior is as described in Section 4.2.1.4.

4.3.4.1 Switch/Tx-Scheduler

Some parameters of the switch and Tx-Scheduler configuration might change by the contents of the alternate structure. Such changes take effect when the alternate structure is enabled.

4.3.4.2 LAN

Expansion ROM code might set LAN QPs for network boot. The sequence of setting a LAN QP is described in Table 10-31 and Section 10.4.3.6.1.

4.3.4.3 Interrupts

Interrupts must be set if used for Admin Queue or LAN queues operation. See Section 9.1.1.1 for the exact flow.

4.4 Driver Load

4.4.1 Introduction

4.4.1.1 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 E810 to initialize it for normal operation. The major initialization steps are:

1. Device driver probes the E810 for resource allocations.
2. Disable interrupts.
3. Initialize the Admin Queue (see Section 9.5.3).
4. Initialize HMC (see Section 9.3.4).
5. Initialize MAC/PHY.
6. Initialize power management - wake-up (see Section 5.3.5).
7. Initialize DCB, including Rx-PB.
8. Initialize the switch and Tx-Scheduler (see Section 7.8.12 and Section 8.3.4.1).
9. Initialize statistics by reading the counter's initial values to serve as a baseline.
10. Init Protocol Engine (see Section 11.5.1).
11. Initialize 1588.
12. Enable interrupts.

At this point, the E810 is ready to initialize VSIs and LAN/PE flows. These are done dynamically during operation:

- Initialize VSI.
- Initialize LAN QP, including its interrupts (see Section 10.4.3 and Section 10.5.5) or Init PE QP (see Section 11.5.1).
- Configure filters (see Section 7.10).

4.4.1.2 Driver Load (SR-IOV)

The E810 approach for virtualized device drivers is to depend heavily on the PF device driver for management of chip resources. The device driver models for newer virtualized operating systems are moving in a direction that requires such functionality. Figure 4-13 shows the high-level initialization flow for virtualized device drivers that use a VF in a guest operating system.

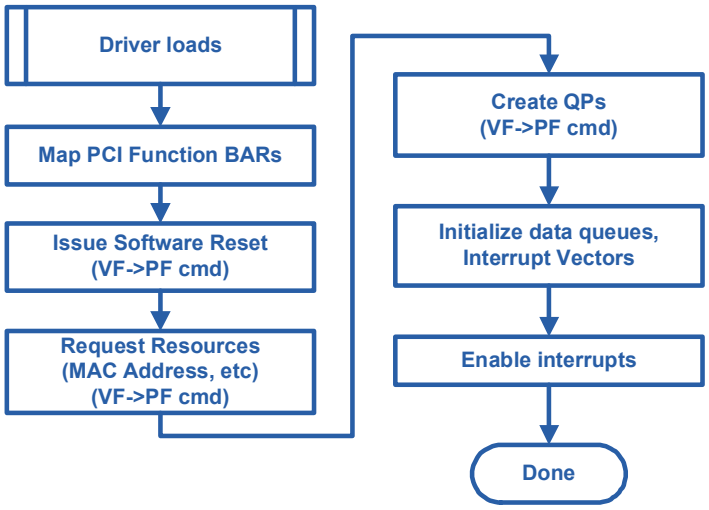


Figure 4-13. Virtualized Device Driver Initialization Flow

4.4.1.2.1 PF Initialization Details

In addition to the regular device driver initialization flow described in Section 4.4.1.1, the PF device driver should apply the following steps to enable support for VMs:

1. After the operating system enables virtual bridging, the PF device driver should create a VEB or a VEPA switching element using the following flow:
   a. Query the switch structure using the Get Switch Configuration command (Section 7.8.12.2.1) to get the initial VSI and initial switch ID associated with the port.
   b. Define mirroring rules using the Mirroring Rules commands (Section 7.8.12.7.1).
   c. Define storm control rules using the Set Storm Control Configuration command (Section 7.8.12.4.1).
   d. Activate malicious driver protection as described in Section 9.2.2.2.1.
2. When the VMM requests that a virtual port be created, the PF driver should:
   a. Create a set of VSIs according to the flow described in the paragraphs that follow. The exact flow depends on the VMM-to-PF API.
   b. Define the bandwidth allocated to the VSI and each of its TCs using the Scheduler Configuration commands (Section 8.3.4.3.6).
   c. If the virtual port is a VF, allow VF access to the network by clearing the associated bit in GL_VIRT_VFLRE register.

4.4.1.2.2 VF Initialization Details

This section describes the flow used to initialize a VF. It refers to various stages shown in Figure 4-13. Only stages involving the hardware are detailed.

4.4.1.2.2.1 Software Reset

A VF software reset can be asserted only by the PF using the VPGEN_VFRTRIG.*VFSWR* field. Following a VF software reset, the VF should request re-initialization of the queues from the PF.

Following the reset, the PF should preform the clean-up steps described in Section 4.1.3.3.

4.4.1.2.2.2 Request Resources and Create/Initialize Data Queues

On top of the resources statically allocated to a VF (interrupts, RSS table, and so on), a VF might have a set of VSIs. Each VSI contains objects such as:

- User priorities (transmit queue groups)
- Queue pairs
- Queuing filters

VSIs can be requested either for LAN or iWARP.

The resources should be requested in the following order (operations with the same stage number can be done in any order):

Step	Resource Requested	PF Action	Rules
1	VSI and UPs	Allocate VSIs according to allocation policies using the Add VSI AQ command (Section 7.8.12.3.1).	The PF should activate the security features in the VSI (anti spoof, port based VLAN) according to the VF settings.
2	Forwarding table entries (MAC and VLANs)	Add the requested filters using the Add Switch Rules command (Section 7.8.12.6.1).	
3	Data queue pairs	Create the requested queue contexts as described in Section 10.4.3 and Section 10.5.5.	
	Protocol Engine resources	The resources for the Protocol Engine are described in Section 11.1 and the PF flow is described in Section 11.5.1.	Before using any Protocol Engine related resources, the Host Memory Cache must be properly programmed. See section Section 9.3 for more details on the Host Memory Cache initialization. Virtual function HMC initialization must be performed by the PF driver on behalf of the VF driver.

4.5 Device/Port/Function Configuration

4.5.1 General

The E810 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.

4.5.1.1 Port-to-Function Mapping

The PFGEN_PORTNUM.*PORT_NUM* per-PF register field (loaded from the NVM) associates each PF with a port.

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 E810 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 E810 provides support for selectively enabling or disabling one or more LAN PCI device(s) in the system.

The E810 provides options to disable its external Ethernet ports and/or PCI functions:

- A strapping pin (DEV_DIS_N) is sampled on LAN_PWR_GOOD 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 LAN_PWR_GOOD and PCIe resets to disable PCI functions. The specific functions that are disabled are determined from the NVM.

Following are 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 E810 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/ LAN_PWR_GOOD/PE_RST_N/in-band reset to the E810 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).

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), all ports are disabled during Dr state. See Section 5.1.3.4.
- 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 PRTGEN_STATUS.*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 E810 is put in a power-down, reset state. Specifically, the E810 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).

4.5.3.1 Dynamic Port Shutdown

Another related, per-port status bit is the PRTGEN_STATUS.*PORT_ACTIVE* indication. This is a dynamic state indicating that the port is temporarily inactive and is powered down. When the port is inactive, reactivating it does not require any reset, and the related PCIe function is still active.

A port is active (PRTGEN_STATUS.*PORT_ACTIVE* = 1b) based on the following rules:

- PRTGEN_STATUS.*PORT_VALID* = 1b, as previously described above AND the E810 is in D0 state (PMCSR.*PowerState* = D0).
  - If link is requested by manageability, the link is kept on. Specifically, hardware ignores disabling the link using the PRTGEN_CNF2.*ACTIVATE_PORT_LINK*.
  - Else,
    - In systems where the application needs to prevent any traffic on link before the device driver is loaded, the PRTGEN_CNF2.*ACTIVATE_PORT_LINK* is loaded from the NVM with the value of 0b (disable).

  - Once the device driver loads, it uses the Set Link and Restart AN with the command bit 2 set (Enable Link). Following this, the EMP enables the link by writing the value of 1b to the PRTGEN_CNF2.*ACTIVATE_PORT_LINK* and starts the link bring-up sequence.
  - If the device driver is about to be removed or disabled, then before going down the device driver might disable the link by the Set link and Restart AN with the command bit 2 cleared (Disable Link). Following this, firmware disables the link by writing the value of 0b to the PRTGEN_CNF2.*ACTIVATE_PORT_LINK*.

- PRTGEN_STATUS.*PORT_VALID* = 1b, as previously described, AND the E810 is in Dr state (PMCSR.*PowerState* = D3).
  - If port is enabled for wake-up. See the description in Section 5.3.
  - Else,
    - The link should be down as it is not required for EMP functionality.
    - BMC might enable manageability by sending the appropriate command (see Section 12.6.3.1). As a result, EMP transitions to pass-through manageability-on state, enables the respective port (if disabled), and starts the link bring-up sequence.
    - If the channel is disabled by the BMC (by sending the appropriate command - see Section 12.6.3.1), and EMP has no other needs for the LAN port in Dr state (like proxy), the EMP reverts to the value of *EMP_LINK_ON* bit in the NVM manageability module or is set to zero (in case of the Disable Channel command with the *ALD* bit set).

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 PFPCI_FUNC.*FUNC_DIS* per-PF bit (loaded from the NVM) disables PCI functions (other than function 0. *FUNC_DIS* for function 0 is hard-wired 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 E810 (such as before the first PCIe configuration cycle to the E810).
  - When a LAN port is enabled or disabled via a Soft SKU command, the EMP identifies (through the PFGEN_PORTNUM registers) the PCI functions associated with the port.
  - EMP then updates the configuration of the functions enabled or disabled by writing to the PFPCI_FUNC.*FUNC_DIS* register bits.
  - PFPCI_FUNC.*FUNC_DIS* for function 0 is hard-wired to enabled, so is not affected by this mechanism.

The result of the previous mechanisms is reflected in the PFPCI_STATUS1.*FUNC_VALID* bit (per PF), which denotes if the function is enabled. A bit is cleared (function disabled) if at least one of the previous mechanisms disables the function.

When a function is disabled:

- It does not respond to PCI configuration cycles (unless specified otherwise). Effectively, the function becomes invisible to the system.
- The *PME_En* bit is cleared to avoid issuing PME.

The Ethernet ports associated with disabled PCI functions are still available for manageability purposes.

4.5.4.1 Dummy Function

When PCI function 0 is disabled, 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 change to other values (dummy function Device ID 0x10A6, Class Code 0xFF0000, with an option to load from the NVM) that claims 4 KB of memory. In addition, the function does not require any I/O space and does not require an interrupt line. All other PCI functions keep their respective locations.

4.5.5 Event Flow for Enable/Disable Ports and PCI Functions

This section describes the expected flow to disable or enable PCI functions or Ethernet ports. Following a power-on reset/LAN_PWR_GOOD/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. 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 E810 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 E810 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/LAN_PWR_GOOD/PE_RST_N/in-band reset.

4.5.5.1 Multi-Function Advertisement

If all but one of the PCI functions are disabled, the E810 is no longer a multi-function device. The E810 normally reports a 0x80 in the PCI configuration header field header type, indicating multi-function capability. However, if only a single LAN is enabled, the E810 reports a 0x0 in this field to signify single-function capability.

4.5.5.2 Legacy Interrupts Use

Each NVM PF configuration module specifies the interrupt line used for each PCI function. When more than one PCI function is enabled, the E810 uses the INTA# to INTD# interrupts for interrupt reporting. The specific interrupt pin used is reported in the *PCI Configuration Header Interrupt Pin* field associated with each PCI function.

However, if only one PCI function is enabled, the INTA# must be used for this PCI function, regardless of the NVM configuration. Under these circumstances, the *Interrupt Pin* field of the PCI header always reports a value of 0x1, indicating INTA# usage.

4.5.5.3 Power Reporting

When more than one PCI function is 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 the LAN Power Consumption NVM word (word 0x22), and is reflected in the *Data* field each time the *Data_Select* field has a value of 0x8 (0x8 = common power value select).

When only one PCI function is enabled, the E810 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 Shared Resource Management

The E810 is a device with multiple external Ethernet ports and that supports multiple PCI functions. Several on-chip resources are shared between device drivers that load the PCI functions exposed by the E810. Additionally, some resources for SR-IOV VFs are managed by the associated PFs. In general, the E810 minimizes the requirement for coordination between device drivers running on different PCI PFs through a combination of resource partitioning and programming interface assistance for remaining resources that are shared.

The E810 handles shared resources using a number of allocation mechanisms and/or access control mechanisms. Table 4-23 lists the supported mechanisms.

Table 4-23. Supported Mechanisms

Class	Description
Dedicated	This resource is associated with a particular E810 element and is always available without respect to any other configuration or setting. Examples of elements in this context could be a PCI function or external Ethernet port.
Administered	Administered resources can be re-assigned based on BMC, BIOS settings, or other OEM specific mechanisms. These resources can be changed with a PCI reset but do not change dynamically after the PCI reset.
Profiles	Resources that are allocated via profiles are typically divided up among different entities based on some combination of other input. A profile might dictate that a resource is divided among the active PCI functions in a particular manner. Another possible usage of a profile is to divide resources based on both the number of PCI functions and the number of external Ethernet ports. The division of resources that are allocated by profiles happens between the time that a PCI reset occurs and might be impacted by an initial device driver load, but resource allocation based on profiles require a PCI reset to change after the initial device driver load.
Pool	Resources that are allocated from pools are typically allocated in a fashion that guarantees no resource starvation to an individual consumer of a resource but enables flexible allocation of remaining resources to any consumer that requires additional resources beyond the minimum. Pools are used for resources that do not need to have a maximum number of resources allocated to all consumers at the same time and that the consumers are reasonably able to fail an allocation.
Service	Resources that are possibly allocated on a per-device or per-port basis but are accessed by a software service that has visibility to multiple device driver interfaces that do not need the E810 to provide an arbitration mechanism. In these cases each PCI function associated with a given resource provides equal access to the resource as other PCI functions associated with the resource. This enables the software service to have the flexibility to access such resources from any device driver instance that it finds available and to switch between device driver instances as the device drivers are stopped and started. Resources that need to accesses directly by a device driver without support from an overlaying software service cannot be handled in this fashion.
Arbitrated	Resources that are allocated on a per-device or per-port basis need an access control mechanism that enables multiple consumers of the resource to operate in an independent manner. The E810 does this by either providing a firmware interface to access the resource where firmware handles the requests one at a time or by other hardware based arbitration scheme.

4.6.1 Resource Profiles

The concept of resource profiles is used in order to make the E810's resource allocation mechanisms easier to understand. Resource profiles dictate how resources listed in Table 4-23 as profiles are distributed among PCI functions so that software drivers do not need to coordinate allocation of these resources.

A resource profile is made up of the set of equations that take input parameters from the system configuration and distributes each resource. Each resource has a different equation for distribution (see the Description column in Table 4-23). Also, the following tables list examples of the result calculations for a full set of resources. Since some of the resource distribution equations simply take the number of PCI functions as input, while others might need additional input (such as the number of external Ethernet ports or the software usage model that a device driver needs to implement), Table 4-24 lists which input parameters effect the allocation of each resource.

Table 4-24. Resource Profiles

Resource	# External Ethernet Ports	# PCI Physical Functions	PCI Function to External Ethernet Port Assignment	# PCI Virtual Functions	Driver Usage Model
LAN Queues	No	Yes	No	Yes	No
MSI-X Vectors	No	Yes	No	Yes	No
Multicast MAC Address Filters	No	Yes	No	No	No
Internal Switching Elements[1]	Yes	Yes	Yes	No	No
Protocol Engine Resources	No	Yes	No	Yes	Yes

1. Resource profiles only impact the default settings for internal switching elements. Run-time switch management software has the ability to override the default number of internal switching components and the internal switch topology.

Table 4-25 lists a few sample sets of typical input parameters that are used with the E810. The initial values of these resources are found in the NVM, but might be overridden by OEM-specific configuration mechanisms followed by a PCI reset.

Note: The device driver usage model can be changed and locked by the first device driver to load following a PCI reset. All other inputs are set at PCI reset.

Table 4-25. Resources per Configuration

Scenario Name	# External Ethernet Ports	# PCI Physical Functions	PCI Function to External Ethernet Port Assignment	# PCI Virtual Functions	Driver Usage Model
Single Port Default	1	1	PF0 -> Port 0	256	Default
Dual Port Default	2	2	PF0 -> Port 0 PF1 -> Port 1	256	Default
Quad Port Default[1]	4	4	PF0 -> Port 0 PF1 -> Port 2 PF2 -> Port 1 PF3 -> Port 3	256	Default
Octal Port Default[2]	8	8	PF0 -> Port 0 PF1 -> Port 1 PF2 -> Port 2 PF3 -> Port 3 PF4 -> Port 4 PF5 -> Port 5 PF6 -> Port 6 PF7 -> Port 7	256	Default
Single Port SR-IOV VF Primary	1	1	PF0 -> Port 0	256	SR-IOV PF Primary
Single Port SR-IOV Even Distribution	1	1	PF0 -> Port 0	256	SR-IOV Even Distribution

1. In this mode, the ports are ordered so that a single 100G port can be used in a 4x25G port configuration. See Table 3-22.
2. See Table 3-21.

Table 4-26 lists the resulting resource distribution for each of the input scenarios listed in Table 4-25. With the exception of the switch topology, none of the resource distribution can be changed after the initial device driver load without a PCI reset occurring. If the resource distribution changes while an active device driver is running, unpredictable results can occur. For example, repartitioning the Host Memory Cache segment descriptor table while a device driver is active causes the E810 to improperly interpret the host memory used for caching resource objects. The E810 provides various mechanisms (such as the HMC resource profile locking) to prevent software from unknowingly causing these types of reconfigurations.

Table 4-26. Resource Distribution

Resource	#LAN QPs Per PF	#LAN QPs Per VF	#MSI-X Vectors Per PF	#MSI-X Vectors Per VF	Default Internal Switch Topology	HMC and Protocol Engine Resources				
						No SR-IOV	SR-IOV Primary Distribution		SR-IOV Even Distribution[1]	
						PF	Per PF	Per VF	Per PF	Per VF
Single Port Default	512	8	129	5	PF0 -> Port 0	4096 SDs 256K QPs 512K CQs 768 CEQs	10 SDs 1024 QPs 2048 CQs 8 CEQs	127 SDs 8160 QPs 16320 CQs 23 CEQs	127 SDs 7943 QPs 15887 CQs 23 CEQs	127 SDs 7943 QPs 15887 CQs 23 CEQs
Dual Port Default	256	8	129	5	PF0 -> Port 0 PF1 -> Port 1	2048 SDs 128K QPs 256K CQs 384 CEQs	10 SDs 1024 QPs 2048 CQs 8 CEQs	127 SDs 8128 QPs 16256 CQs 23 CEQs	120 SDs 7710 QPs 15420 CQs 23 CEQs	120 SDs 7710 QPs 15420 CQs 23 CEQs
Quad Port Default	128	8	129	5	PF0 -> Port 0 PF1 -> Port PF2 -> Port PF3 -> Port 3	1024 SDs 64K QPs 128K CQs 192 CEQs	10 SDs 1024 QPs 2048 CQs 8 CEQs	126 SDs 8064 QPs 16128 CQs 23 CEQs	113 SDs 7281 QPs 14562 CQs 23 CEQs	113 SDs 7281 QPs 14562 CQs 23 CEQs
Octal Ports Defaults	64	8	129	5	PF0 -> Port 0 PF1 -> Port 2 PF2 -> Port 4 PF3 -> Port 6 PF4 -> Port 7 PF5 -> Port 5 PF6 -> Port 3 PF7 -> Port 1	512 SDs 32K QPs 64K CQs 96 CEQs	10 SDs 1024 QPs 2048 CQs 8 CEQs	125 SDs 7936 QPs 15872 CQs 22 CEQs	102 SDs 6553 QPs 13107 CQs 19 CEQs	102 SDs 6553 QPs 13107 CQs 19 CEQs

1. The SR-IOV Even Distribution is shown with 16 VFs enabled for Protocol Engine operation.

Chapter 5 Power Management

This section defines how Power Management is implemented in the E810.

5.1 PCIe Power Management

5.1.1 Auxiliary Power Usage

Auxiliary power can be used for powering the E810 while the system is in low power state ($D3_{cold}$). The E810 uses the AUX_PWR pin as an indication that auxiliary power is available to it. The E810 uses this indication to advertise $D3_{cold}$ wake-up support in the PMC.*PME_Support* field and to set 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 E810 only uses the auxiliary power if one of the functions requiring it (as indicated below) are enabled (and, if auxiliary power is available). If the usage of auxiliary power is not enabled (that is, none of the functions requiring it are enabled), the E810 must be power-gated and does not consume auxiliary power in $D3_{cold}$.

If AUX power usage during $D3_{cold}$ is supported (as indicated below), all of the E810's sticky bits preserve their values and only get reset by the power-up reset (detection of power rising).

When AUX power is applied to the E810, the actual usage of AUX power during $D3_{cold}$ is controlled through the following factors:

- NVM loaded bits:
  - Manageability needed — Bit per port loaded from NVM, which indicates if the relevant port link should be established for EMP functionality. The E810 provides the proper clocking to establish the required port link/s at this state.
  - *APME* — Bit per function (loaded to PFPM_APM[PF]) indicates if the relevant PF associated to a port link should be established to enable APM WoL. The E810 provides the proper clocking to establish the required port link/s at this state.
- PCIe configuration bits:
  - *PME_En* — This bit in the PMCSR PCI config space is used to determine if the E810 is allowed to consume AUX power for WoL.
- EMP firmware decision:
  - The EMP firmware might prevent the device from being power-gated in Sx states, and thus consume auxiliary power. The E810 should be specifically configured by the EMP firmware to provide the proper clocking or to establish the required port link/s at this state.

The following pseudo code defines AUX Power usage where *APME* and *PME_En* bits refer to a logical OR over all the PFs attached to the port:

```
If (AUX_PWR = 1),
    If ((APME or EMP_LINK_ON or PME_En) = 1),
        AUX power is used to preserve link functionality.
    Else,
        Link functionality is not preserved during AUX power supply.
    End-IF
End-If
```

5.1.2 PCIe Link Power Management

The PCIe link state follows the power management state of the device. Since the E810 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 E810 supports all PCIe power management link states:

- L0 state is used in D0u and D0a states.
- The L0s state is used in D0a and D0u states each time link conditions apply.
- 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.

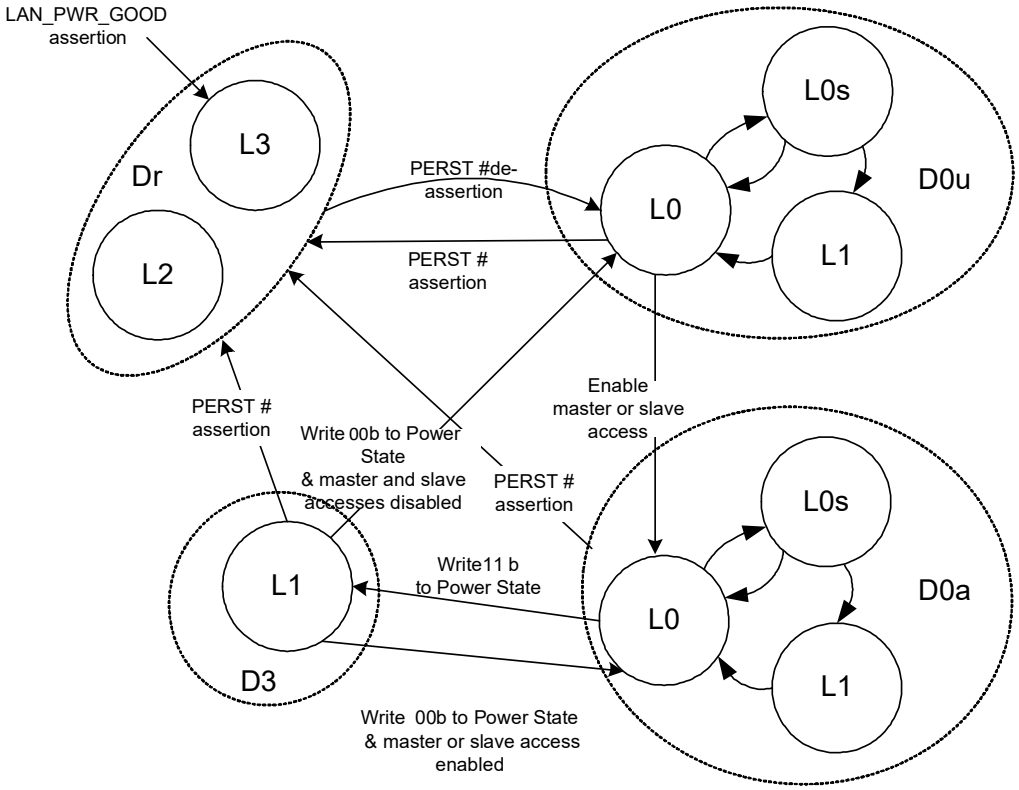


Figure 5-1. Link Power Management State Diagram

While in L0 state, the E810 transitions the transmit lane(s) into L0s state once the idle conditions are met for a period of time defined as follows.

- L0s configuration fields are:
  - L0s enable — The default value of the *Active State Link PM Control* field in the PCIe Link Control register is set to 00b (both L0s and L1 disabled). System software can later write a different value into the Link Control register.
  - L0s exit latency (as published in the *L0s Exit Latency* field of the Link Capabilities register) is loaded from/to the GLPCI_PMSUP.*L0S_EXIT_LAT* NVM/register field. Separate values are loaded when the E810 shares the same reference PCIe clock with its partner across the link, and when the E810 uses a different reference clock than its partner across the link. The E810 reports whether it uses the slot clock configuration through the *PCIe Slot Clock Configuration* bit loaded from/to the GLPCI_PMSUP.*SLOT_CLK* NVM/register bit.
  - L0s acceptable latency (as published in the *Endpoint L0s Acceptable Latency* field of the Device Capabilities register) is loaded from the GLPCI_PMSUP.*L0S_ACC_LAT* NVM/register field.

The E810 transitions the link into L0s state once the PCIe link has been idle for a period of time defined in the *l0s_idle_timer* field of PMIDLTMR register loaded from RO PCIe LCB module in NVM. The E810 then transitions the link into L1 state once the PCIe link has been in L0s state for a further period as defined in the *l1_idle_timer* register field of the same register.

The following NVM fields control L1 behavior:

- *L1 support* — Indicates support for ASPM L1 in the PCIe configuration space (loaded into the *Active State Link PM Support* field).
- *L1 exit latency* — Defines L1 active exit latency. The default value is loaded from/to the GLPCI_PMSUP.*L1_EXIT_LAT* NVM/register field.
- *L1 acceptable latency* — Defines L1 active acceptable exit latency. The default value is loaded from/to the GLPCI_PMSUP.*L1_ACC_LAT* NVM/register field.

5.1.3 Power States

The E810 supports the D0 and D3 architectural power states as described earlier. Internally, the E8100 supports the following power states:

- **D0u ($D0_{un\text{-}initialized}$)** — An architectural sub-state of D0.
- **D0a ($D0_{active}$)** — An architectural sub-state of D0.
- **D3** — Architecture state D3hot.
- **Dr** — Internal state that contains the architecture $D3_{cold}$ state. Dr state is entered when PE_RST_N is asserted, a PCIe in-band reset is received, or if the function is disabled.

Figure 5-2 shows the power states and transitions between them.

Note: PCI resets (cold/warm and hot) are reflected to the E8100 as primary bus reset (in case of cold/warm reset) or secondary bus reset (in case of a hot reset).

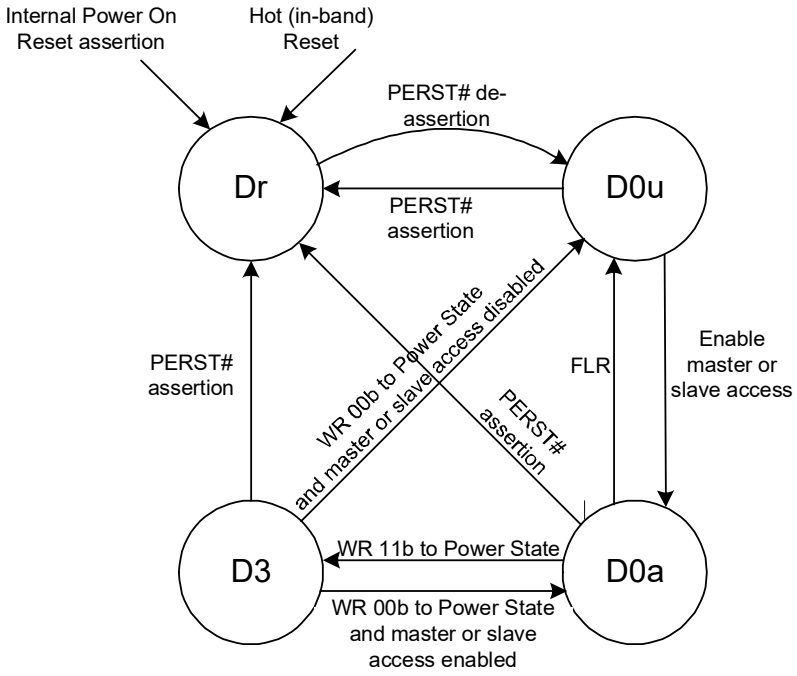


Figure 5-2. Power Management State Diagram

5.1.3.1 $D0_{uninitialized}$ (D0u) State

The D0u state is an architectural low-power state. In this state, the device is out of reset (conventional and FLR) but did not complete the enumeration and configuration stage.

5.1.3.1.1 Entry to a D0u State

D0u is reached from either the Dr state (on de-assertion of internal PE_RST_N and auto-load of hardware configuration done) or the D3hot state (by configuration software writing a value of 00b to the *Power State* field of the PCI PM registers while master and slave accesses are disabled). De-assertion of internal PE_RST_N causes the entire state of the E810 to be cleared except for bits defined as sticky in configuration space. PCIe configuration is loaded from the NVM, followed by establishment of the PCIe enumeration. Once this is done, configuration software can access the E810.

5.1.3.2 $D0_{active}$ (D0a) State

A E810 Function enters the $D0_{active}$ state whenever any single or combination of the function's *Memory Space Enable*, *I/O Space Enable*, or *Bus Master Enable* bits are enabled by system software in the PCI Command configuration register.

In this 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.

Notes:

- Wake behavior:
  - An APM wake event (PM_PME message) due to reception of a Magic Packet is not generated when the function is in $D0_{active}$ state.
  - Any APM wake up previously active remains active when moving from D3 to D0. That is, APM enable is loaded from NVM and is not reset by any reset but POR.
  - WAKE# pin is never toggled for an APM wake event when a function is in D0.
- If APM wake is required in D3, software driver should not disable APM wake-up via clearing the PFPM_APM.*APME* bit on entry into D0. Otherwise, APM wake following a system crash and entry into S3, S4, or S5 system power management state are not enabled. Following entry into D0, software device driver can activate the wake-up filters by writing to the Wake Up Filter Control (PFPM_WUFC) register.

5.1.3.2.1 Entry to a D0a State

D0a is entered from either the D0u state (by writing a 1b to the *Memory Access Enable* or the *I/O Access Enable* bit, or the *BME* bit of the PCI Command configuration register) or from the $D3_{hot}$ state (by configuration software writing a value of 00b to the *Power State* field of the PCI PM configuration registers while master or slave accesses is enabled). The receive and transmit flows of the appropriate LAN function are enabled.

5.1.3.3 D3 State (PCI-PM $D3_{hot}$)

The E810 functions can transition to D3 when the system writes a 11b to the *Power State* field of the Power Management Control/Status Register (PMCSR). Any wake-up filter settings that were enabled before entering this state are maintained. The E810 does not clear any bit in the PCIe configuration space of a function during the function's transition to D3 state. While in D3, the appropriate function of the E810 does not generate master cycles.

Configuration and message requests are the only TLPs accepted by a function in the $D3_{hot}$ state. All other received requests must be handled as unsupported requests, and all received completions are handled as unexpected completions. If an error caused by a received TLP (such as an unsupported request) is detected while in $D3_{hot}$ and reporting is enabled, the link must be returned to L0 if it is not already in L0, and an error message must be sent. See section 5.3.1.4.1 in the *PCIe Base Specification*.

5.1.3.3.1 Entry to D3 State

Transition to D3 state is through a configuration write to the *Power State* field of the PMCSR PCIe configuration register.

Prior to transition from D0 to the D3 state, the device driver disables scheduling of further tasks to the E810. It masks all interrupts, and it does not write to the Transmit Descriptor Tail register or to the Receive Descriptor Tail register.

If wake up capability is needed, system software should enable wake capability by setting to 1b the *PME_En* bit in the PMCSR PCIe configuration register of the PF. After Wake capability is enabled the software device driver should set up the required wake up functionality using the flow described in Section 5.3.5 prior to the D3 transition.

Notes:

- The PMCSR.*PME_En* bit setting can be overridden via the PFPM_APM.*APME* bit.
- If operation during D3*cold* is required, even when wake capability is not required (for example, for manageability operation), system should also set the *Auxiliary (AUX) Power PM Enable* bit in the PCIe Device Control register.

If all PCI functions are programmed into D3 state, the E810 attempts to brings its PCIe link into the L1 link state. As part of the transition into L1 state, the E810 suspends scheduling of new host TLPs, MCTP over PCIe VDMs is not suspended. The E810 waits for the completion of all previous TLPs it has sent. Any receive packets that have not been transferred into system memory are kept in the device (and discarded later on D3 exit). Any transmit packets that have not be sent can still be transmitted (assuming the Ethernet link is up).

In preparation for a possible transition to $D3_{cold}$ state, the device driver might disable up to all but one of its LAN ports (LAN disable) and/or transition the remaining link(s) to GbE speed (if supported by the network interface).

5.1.3.3.2 Exit from D3 State

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 $D3_{cold}$ state). To transition back to D0u, the system writes a 00b to the *Power State* field of the Power Management Control/Status Register (PMCSR). Transition to Dr state is through PE_RST_N assertion.

The *No_Soft_Reset* bit in the PCIe Power Management Control/Status (PMCSR) register in the E810 is always set to 1b to indicate that the E810 does not perform an internal reset on transition from $D3_{hot}$ to D0, so that transition does not disrupt the proper operation of other active functions. Software is not required to re-initialize the function's configuration space after a transition from $D3_{hot}$ to D0 (the function is in the D0 state).

The function is reset if the Link state had transition to the L2/L3 Ready state, on transition from $D3_{cold}$ to D0, if FLR is asserted, or if transition $D3_{hot}$ to D0 is caused by assertion of PCIe reset (PE_RST pin).

5.1.3.4 Dr State ($D3_{cold}$)

Transition to Dr state is initiated on system power-up. Dr state begins with the assertion of the internal power detection circuit (LAN_PWR_GOOD) and ends after the completion of the NVM auto-load.

Any wake-up settings that were enabled before entering this reset state are maintained.

The system can 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 E810 can maintain functionality (for WoL or manageability) or can enter a Dr Disable state (if no WoL and no manageability) for minimal device power. The Dr Disable mode is described in Section 5.1.3.4.1.

5.1.3.4.1 Dr Disable Mode

The E810 enters a Dr disable mode on transition to $D3_{cold}$ state when it does not need to maintain any functionality. The conditions to enter either state are:

- The device (all PCI functions) is in Dr state.
- APM WOL is inactive for all PCI functions.
- Pass-through manageability is disabled.

Entry into Dr disable is usually done on assertion of PCIe PE_RST_N. It can 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 device 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 device 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 can remove the E810 power. If the platform removes the E810 power, it must remove all power rails from the device if it needs to use this capability. Exit from this state is through power-up cycle to the E810.

Note: The state of the TX_DIS (Optical module Tx disable connected to SDP pins) or any of the other SDP pins is undefined once power is removed from the device.

5.1.3.4.2 Entry to Dr State

Dr entry on platform power-up is as follows:

- Assertion of the internal power detection circuit (LAN_PWR_GOOD). Device power is kept to a minimum by keeping the network interfaces in low power.
- The NVM is then read and determines device configuration.
- If the *APM Enable* bit in the NVM is set, APM wake up is enabled (for each port independently).
- If the *MNG Enable* bit in the NVM word is set, pass-through manageability is enabled.
- Each of the LAN ports can be enabled if required for WoL or manageability. See Section 5.2 for exact condition to enable a port.
- 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 through assertion of 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 a user selected a shut down operating system option), in which case the device driver can intervene. Or, it can be an emergency transition (like power button override), in which case, the device driver is not notified.

Transition from D3 state to Dr state is done by assertion of 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.

5.1.3.5 Protocol Engine Power Save Modes

Internally, the E810 supports two modes to reduce component power opportunistically by disabling the internal RDMA Protocol Engine (PE) via PE Clock Gating.

If the RDMA functionality is available, but not activated by software (for example, when PCIe reset is asserted), the E810 supports clock gating the PE block. Hardware determines whether RDMA is enabled if the GLHMC_PEHTEOBJSZ register is programmed to any non-zero value. GLHMC_PEHTEOBJSZ is reset by CORER. As with power gating, register accesses which would be decoded to the PE block are handled gracefully. Write transactions are silently discarded, and read accesses return deterministic values (0xDEADBEEF).

5.2 Network Interfaces Power Management

The E810 disables a network interface and transitions the interface into low-power state in the following cases:

- All LAN ports are in LAN disable mode as a result of the DEV_DIS_N pin.
- Low power state functionality as described in Section 5.1.1.

When the E810 is in low-power state, it asserts the respective TX_DIS pin (connected to an SDP pin) to enable powering down an external PHY or optical module as well.

5.2.1 Energy Efficient Ethernet (EEE)

Note: Energy Efficient Ethernet (EEE) is not currently supported in the E810. Some references may remain in this document in the event that it is enabled in the future.

5.2.2 Low Power Link Up (LPLU)

Auto-negotiation enables establishment of link speed at the Highest Common Denominator (HCD). When all PCIe functions belonging to a port are in Dx low power state, and the E810 is connected to an auxiliary power supply and main power is down, power saving might be more important than performance.

The E810 supports a mode where it auto-negotiates to the Lowest Common Denominator (LCD) link speed (that is, the lowest commonly-supported speed) in low power states. Initially, in this mode, the E810 attempts to negotiate to the lowest link rate. If the link partner does not support the lowest link rate, the E810 auto-negotiates to the lowest link rate advertised by the link partner.

5.2.2.1 LPLU-Related Link Speed Change

Figure 5-3 describes the flow for changing the link speed when entering/existing LPLU state.

The E810 attempts to lower the link to the lowest commonly-supported speed when entering to LPLU state. Entering LPLU state, and switching to a lower link speed is done by clearing the relevant bits in the auto-negotiation advertised capabilities in the (internal/external) PHY configuration (advertising only the lowest common speed), and triggering an auto-negotiation process. The device is expected to link at lower speed as higher speeds are not advertised as supported.

Similarly, exiting LPLU state and switching to a higher link speed is done by setting the relevant bits in the auto-negotiation advertised capabilities in the PHY configuration, and restarting auto-negotiation. The device is now expected to link at the highest common speed.

Note: Given the above, a temporary link loss (and a disruption to traffic) is expected when entering/exiting LPLU state.

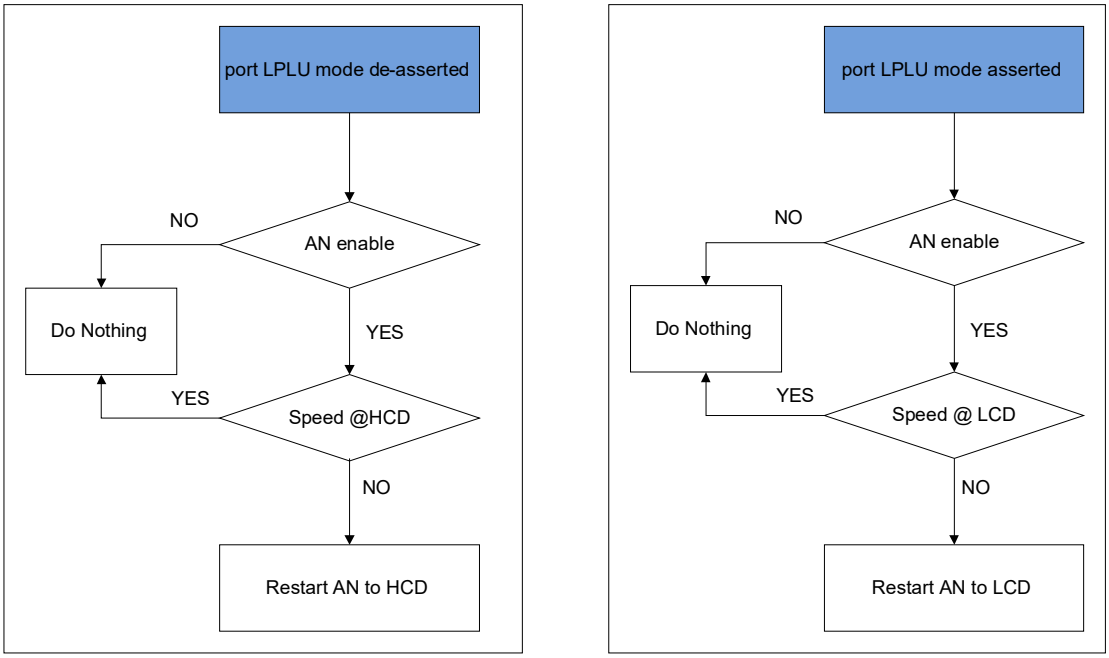


Figure 5-3. Link Speed Change on LPLU Assert/Clear

5.3 Wake-Up

The E810 supports wake up functionality in accordance with the APM and ACPI specifications. The following list provides a high-level view of the functionality supported in each specification, while the following sub-section provide more details and elaboration.

- Advanced Power Management (APM) wake-up (enabled via PFPM_APM.*APME*), which means support for wake from S5 (Soft Off) states using the well-known Magic Packet wake packet format.
- ACPI wake-up capabilities (enabled via PMCSR.PME_En):
  - Link State Change (*LNKC*) wake-up, which means the ability to wake up the host from S5 when a LAN link state goes up.
  - Magic Packet (*MAG*) wake-up, which means support for wake from S5 (Soft Off) states using the well-known Magic Packet wake packet format.
  - Firmware Reset (*FW_RST_WK*) wake-up, which means the ability to wake up the host from S5 when an EMPR event occurs.
  - Manageability (*MNG*) wake-up, which means the ability for the EMP to wake up the host from S5, on its own initiative. No usage model is actually defined for this option.

These wake-up mechanisms are basically controlled by the PF via the corresponding bit in the PFPM_WUFC and PFPM_WUS registers.

5.3.1 Advanced Power Management Wake-Up

This feature has existed in the 10/100 Mb/s NICs for several generations. System was activated from S3, S4, or S5 low power states on reception of a Magic Packet, which is a broadcast or unicast packet with an explicit data pattern. On reception of a Magic Packet, the E810 can wake up the system by also asserting the PCIe PE_WAKE_N signal and optionally via an in-band PM_PME message.

APM wake-up operation is controlled by the per-PF APM Control (PFPM_APM) register. At power-up, the E810 reads the *APM Enable* bit from the NVM into the APM Enable (PFPM_APM.*APME*) bit. This bit is used to enable WoL by overriding the *PME_En* bit. This bit controls enabling of the APM wake-up. Software enables or disables APM wake-up using the flows described in Section 5.3.5.1 and Section 5.3.5.2, respectively.

When APM wake-up is enabled, the E810 checks all incoming packets passing L2 filters (BCST, MCST, UCST) for Magic Packets. See Section 5.3.3.1.1 for a definition of Magic Packets. BCST and UCST (according to PFPM_SAL/H that are loaded from NVM and properly assigned by firmware to the PRTPM_SAL/H registers) Magic Packet filtering is enabled by default. To enable promiscuous MCST Magic Packets, WoL PRTPM_SAH.*MC_MAG_EN* must be set to 1b.

Once the E810 receives a matching Magic Packet, and the PFPM_APM.*APME* bit is set to 1b, it executes the ACPI flow described in Section 5.3.5.3.

Note: When the PFPM_APM.*APME* bit is set, a wake event is issued (PE_WAKE_N pin is asserted and a PM_PME PCIe message is issued), even if the PMCSR.*PME_En* bit in configuration space is cleared. To disable system wake-up when PMCSR.*PME_En* is cleared, software driver should clear the PFPM_APM.*APME* bit after power-up or PCIe reset.

5.3.2 ACPI Power Management Wake-Up

The E810 supports ACPI power management-based wake-up. It can generate system wake-up events from a number of sources:

- Detection of a change in network link (cable connected or disconnected).
- Firmware Reset (*FW_RST_WK*) wake-up, which means the ability to wake up the host from S5 when an EMPR event occurs.
- Magic Packet (*MAG*) wake-up, which means support for wake from S5 (Soft Off) states using the well-known Magic Packet wake packet format.
- Manageability (*MNG*) wake up, which means the ability for the EMP to wake up the host from S5, on its own initiative. No usage model is actually defined for this option.

Note: ACPI wake-up operation is controlled by the per-PF Wake Up Filter Control (PFPM_WUFC) register, and wake-up status is reported in the per PF Wake Up Status (PFPM_WUS) register. In the PCIe configuration space, the PMCSR register supports wake up. Software enables ACPI wake-up using the flow described in Section 5.3.5.

5.3.3 Wake-Up Filters

The E810 supports issuing wake-up indication to the Host on reception of non-errored packets when the device is in D3 low power state or in Dr with WoL enabled, using Magic Packet filters set from NVM.

Support of wake-up in Dr is only if the *AUX_PWR* bit is asserted.

5.3.3.1 Wake-Up Filter Types

Magic Packets are detected per-PF by the E810 as a possible wake-up.

5.3.3.1.1 Magic Packet

Magic Packets are defined in http://support.amd.com/TechDocs/20213.pdf as:

> "Once the LAN controller has been put into the Magic Packet mode, it scans all incoming frames addressed to the node for a specific data sequence. This sequence indicates to the controller that this is a Magic Packet frame. A Magic Packet frame must also meet the basic requirements for the LAN technology chosen, such as Source Address, Destination Address (which can be the receiving station's IEEE address or a Multicast address which includes the Broadcast address), and CRC. A Magic Packet must also be a valid non error L2 packet. The specific data sequence consists of 16 repetitions of the IEEE address of this node, with no breaks or interruptions. This sequence can be located anywhere within the packet, but must be preceded by a synchronization stream. The synchronization stream allows the scanning state machine to be much simpler. The synchronization stream is defined as 6 bytes of 0xFF. The device will also accept a Broadcast frame, as long as the 16 repetitions of the IEEE address match the address of the machine to be awakened."

The E810 expects the destination address to either:

- Be the broadcast address (FF.FF.FF.FF.FF.FF).
- Match the value of the port L2 Ethernet destination MAC Address (DA) set by firmware to the relevant filters PRTPM_SAL/H.

On power-up, the firmware loads the PRTPM_SAL/H addresses from NVM, each address entry contains:

- MAC Address for Magic Packet filtering.
- Address Valid (AV) indication.
- Promiscuous Multicast magic filtering enable (MC) indicating the L2 destination address can be any multicast address.
- The PF number (*PF_NUM*) to be reported if a Magic Packet matches this filter.

The firmware uses the PFGEN_PORTNUM registers to map the per PF MAC Address to the relevant ports.

The E810 searches for the contents of the port's Ethernet Destination MAC Address Receive Address as the embedded IEEE address. It considers any non-0xFF byte after a series of at least 6 0xFFs to be the start of the IEEE address for comparison purposes. For example, it catches the case of 7 0xFFs followed by the IEEE address). As soon as one of the first 96 bytes after a string of 0xFFs do not match, it continues to search for another set of at least 6 0xFFs followed by the 16 copies of the IEEE address later in the packet.

Note: This definition precludes the first byte of the destination address from being FF.

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 broadcast packet reception is not enabled. If APM wake-up (wake-up by a Magic Packet) is enabled in the NVM, the E810 starts up with the Ethernet MAC Destination Address loaded from the NVM. This enables the E810 to accept packets with the matching IEEE address before the software device driver loads.

Table 5-1. Magic Packet Structure

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	T=(0/4/8)	Possible STag		Skip	
12 + T	S=(0/4)	Possible VLAN Tag		Skip	
12 + T + S	D=(0/8)	Possible Length + LLC/SNAP Header		Skip	
12 + T+ 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 relevant Station Addresses (PRTPM_SAH, PRTPM_SAL) registers.

5.3.4 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 (PF) 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 per VF.

5.3.5 Wake-Up Flows

5.3.5.1 Wake-Up Enable Flow

A WoL register set exists in each PF. Each set consists of the following wake-up filters and registers:

- PFPM_WUC — Wake Up Control Register.
- PFPM_WUS — Wake Up Status Register.

On power-up, values loaded from the NVM into the following per PF register define enablement of APM wake-up functionality:

- The NVM *APM Enable* bit is loaded to the PFPM_APM.*APME* register bit to enable detection of a Magic Packet destined to the PF.

Note: The *EN_APM_D0* bit also enables the wake functionality in ACPI mode while the device is in D0. Therefore, if any of the ACPI wake filters are enabled (See Section 5.3.2) and wake events in DO are not desired, *EN_APM_DO* must be cleared by the software. Similarly, if software wishes to enable wake events in D0, the *EN_APM_D0* bit must be set by the software in addition to enabling the relevant ACPI wake filters.

5.3.5.2 Wake-Up Disable Flow

Once the E810 wakes the system following transition to D0, the software driver can disable Wake-on-LAN directly by:

1. Clearing all the pending wake-up status bits in the Wake Up Status (PFPM_WUS) register.
2. Clearing the relevant bits in the PFPM_WUC register.

Note: This flow is not used by regular Intel drivers.

5.3.5.3 ACPI Wake-Up Flow

Once wake up is enabled, the E810 monitors the enabled wake functions. If at least one of the enabled wake events occurs, the E810 does the following:

- Sets the *PME_Status* bit in the PFPM_WUS and PCI PMCSR registers.
- Sets the proper bit, corresponding to the wake reason in PFPM_WUS:
  - *LINKC* for link status change.
  - *MAG* for Magic Packet reception.
  - *MNG* for general manageability wake-up.
  - *FW_RST_WK* for EMP reset event.
- If the *PME_En* bit in the PMCSR configuration register is set, asserts PE_WAKE_N and/or sends a PM_PME PCIe message.

Note: The PE_WAKE_N signal remains asserted, PM_PME Messages are periodically sent to the host, and the E810 ignores any subsequent ACPI wake-up packets and link change events for that function 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, or until de-assertion of PERST.

NOTE: ***This page intentionally left blank.***

Chapter 6 Non-Volatile Memory Map

6.1 NVM Organization

Note: All the offsets in this chapter are relative to the E810 Configuration region. This region is located in the flash at offset 4 KB as described in Section 3.4.2.1. 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	12K	A set of TLV sections as described in section TBD containing all the information that should be preserved across updates.
Shadow RAM - Non-Preserved	Auto-Load Sections	Variable	Sections containing the auto-loaded 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	32K	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		Signed section containing the Option ROM image.
	OROM Section - Second Copy		A second copy of the OROM section for updates.
	Extended TLVs - 1st Copy		Extension of the PFA module.
	Extended TLVs - 2nd Copy		
	Scratch Pad Area		Areas used for data not double banked.

6.1.2 NVM Header

Table 6-2 lists the format of the NVM Header section. It includes words with specific content and pointers to the first level of NVM modules.

- Items which are pointers to modules end with the pointer in there name.
  - The pointer to non-Shadow RAM modules that 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 "For Section [Auth] 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.

  A read-only item can be written only when in the blank Flash programming mode.
- 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 (Bit 15) 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/Word Name	For Section [Auth]	Accessed by	Loading Trigger	CSR Format	Module is in Shadow RAM
0x00	NVM Control Word 1		EMP	POR		
0x01	Non-Persistent End Pointer		EMP		No	Yes
0x02	Last PFA Word Pointer				No	Yes
0x03 - 0x05	Reserved					
0x06	RO PCIR Registers Auto-Load Module Pointer	Auth	EMP	PCIR	Yes	Yes
0x07	Auto Generated Pointers Pointer	Auth	EMP/SW	POR	No	Yes
0x08	Reserved					
0x09	EMP Global Module Pointer	Auth	EMP	GLOBR	No	No
0x0A	Guarded Zone Pointer	Hidden from Software	EMP		No	No
0x0B	EMP Image Pointer	Auth	EMP	EMPR	No	No
0x0C	PE Image Pointer	Auth	PE	CORER	No	No
0x0D	Reserved					
0x0E	Manageability Module Pointer	Adaptive	EMP	EMPR	No	Yes
0x0F	EMP Settings Module Pointer	Auth	EMP	CORER/EMPR	No	No
0x10	SW Compatibility Word 1		SW/BIOS	POR		
0x11	SW Compatibility Word 2		SW/BIOS	POR		
0x12	SW Compatibility Word 3		SW/BIOS	POR		
0x13	SW Compatibility Word 4		SW/BIOS	POR		
0x14	SW Compatibility Word 5		SW/BIOS	POR		
0x15	PBA Flags		SW/BIOS	POR		
0x16	PBA Block Pointer	Auth	SW/BIOS	POR	No	Yes
0x17	Reserved					
0x18	Software Reserved Word 1 - Dev Starter Revision		SW/BIOS/EMP	POR		

Table 6-2. NVM Header Map [continued]

Word Address	Pointed Module/Word Name	For Section [Auth]	Accessed by	Loading Trigger	CSR Format	Module is in Shadow RAM
0x19	Software Reserved Word 2		SW/BIOS	POR		
0x1A	Software Reserved Word 3		SW/BIOS	POR		
0x1B	Software Reserved Word 4		SW/BIOS	POR		
0x1C	Software Reserved Word 5		SW/BIOS	POR		
0x1D	Software Reserved Word 6		SW/BIOS	POR		
0x1E	Software Reserved Word 7		SW/BIOS	POR		
0x1F	Software Reserved Word 8		SW/BIOS	POR		
0x20	Software Reserved Word 9		SW/BIOS	POR		
0x21	Software Reserved Word 10		SW/BIOS	POR		
0x22	Software Reserved Word 11		SW/BIOS	POR		
0x23	Software Reserved Word 12		SW/BIOS	POR		
0x24	Software Reserved Word 13		SW/BIOS	POR		
0x25	Software Reserved Word 14		SW/BIOS	POR		
0x26	Software Reserved Word 15		SW/BIOS	POR		
0x27	Software Reserved Word 16		SW/BIOS	POR		
0x28	Software Reserved Word 17		SW/BIOS	POR		
0x29	Software Reserved Word 18 - Map Version		SW/BIOS	POR		
0x2A	Software Reserved Word 19 - NVM Image Version		SW/BIOS	POR		
0x2B	Software Reserved Word 20 - NVM Structure Version		SW/BIOS	POR		
0x2C	Software Reserved Word 21					
0x2D	Software Reserved Word 22 - EETRACK ID 1		SW/BIOS	POR		
0x2E	Software Reserved Word 23 - EETRACK ID 2		SW/BIOS	POR		
0x2F - 0x32	Reserved					
0x33	IBA Capabilities		BIOS	POR		
0x34	Software Reserved Word 24 - Original EETRACK ID 1		BIOS	POR		
0x35	Software Reserved Word 25 - Original EETRACK ID 2		BIOS	POR		
0x36 - 0x3A	Reserved					
0x3B	GLOBR Registers Auto-Load Pointer	Adaptive	EMP	GLOBR	Yes	Yes
0x3C	CORER Registers Auto-Load Pointer	Adaptive	EMP	CORER	Yes	Yes
0x3D	PHY Configuration Scripts Pointer	Auth	EMP	POR	No	No
0x3E	Reserved					

Table 6-2. NVM Header Map [continued]

Word Address	Pointed Module/Word Name	For Section [Auth]	Accessed by	Loading Trigger	CSR Format	Module is in Shadow RAM
0x3F	Reserved (was: Software Checksum) - moved to PFA					
0x40	Preserved Field Area Pointer	RO	SW/EMP	POR	No	Yes
0x41	Reserved					
0x42	1st NVM Bank pointer	RO	EMP	N/A	No	No
0x43	NVM Bank Area Size		EMP			
0x44	1st OROM Bank pointer	RO	EMP	N/A	No	No
0x45	OROM Bank Area Size	RO	EMP			
0x46	1st TLV Extension Bank Pointer	RW	EMP			
0x47	TLV Extension Bank Area Size		EMP			
0x48	EMP SR Settings Pointer	Adaptive	EMP	EMPR		Yes
0x49	Reserved					
0x4A	PE CORER Registers Auto-Load Pointer			CORER[1]		
0x4B	Link Topology Scratch Pad Area Pointer	RW	SW	POR	No	No
0x4C	Link Topology Scratch Pad Area Size		SW	POR		
0x4D	Configuration Metadata Pointer	Auth	EMP	EMPR	No	No
0x4E - 0x4F	Reserved					
0x50	FW Scratch Pad Area Pointer	RW	EMP		No	No
0x51	FW Scratch Pad Area Size		EMP		No	
0x52	Reserved					
0x53	PE Settings Module Pointer	Auth	PE	CORER	No	No
0x54	PHY Core Post PLL Configuration Module Pointer	Auth	EMP	POR	No	No
0x55	Soft SKUs		EMP		No	
0x56	External CORER Registers Auto-load Pointer	Auth	EMP	CORER	Yes	No
0x57	Reserved					
0x58	DCB Rx Module Pointer	Auth	EMP			No
0x59 - 0x5A	Reserved					
0x5B	DCB Rx Module Pointer	Auth	EMP	CORER	Yes	No
0x5C	DCB Tx Module Pointer	Auth	EMP	CORER	Yes	No
0x5D - 0xFF	Spare NVM Header Words			*POR*		

1. Loaded after a CORER and Protocol Engine is enabled.

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 E810. In Table 6-2, these modules are referred to 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. 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. 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.

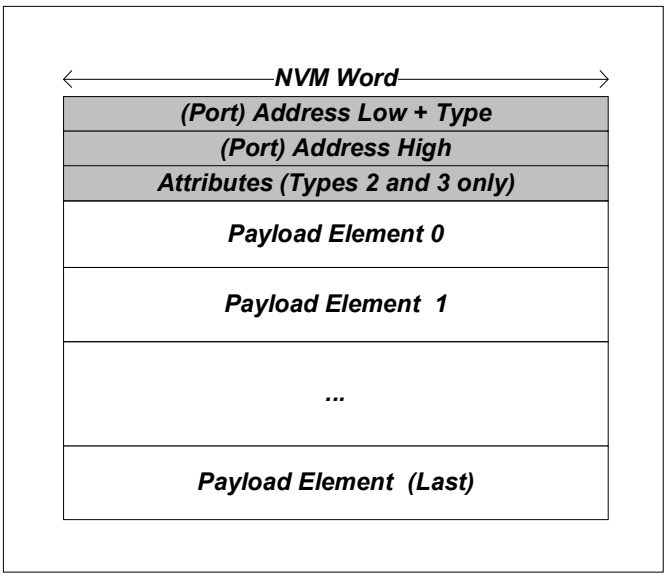


Figure 6-1. Structure of a Flexible Functionality Module

Following are descriptions 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 and 2), or as an indirect address referred to as a port address (Type 3 and 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-bits wide.
    - 001b = Data is 64-bits wide.
    - 010b = Data is 128-bits wide.
    - 011b = Data is 256-bits 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 E810. 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.

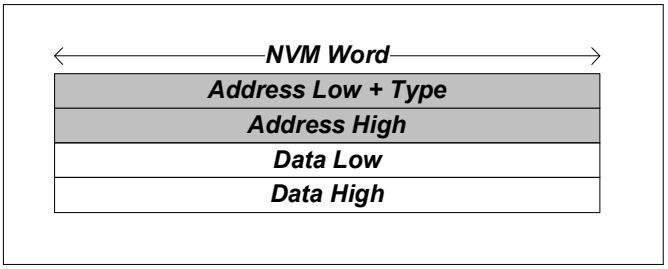


Figure 6-2. Structure of a Type 1 Module

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

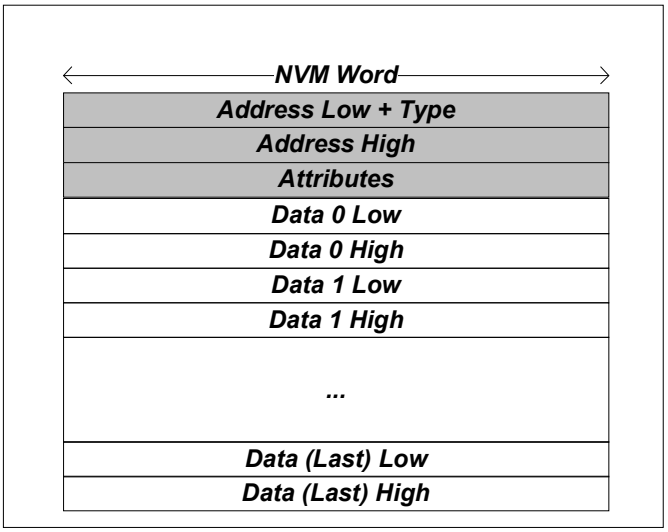


Figure 6-3. Structure of a Type 2 Module

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 the E810) with each write of a new data element from the list. Consecutive data elements are written into the port.
- The *Data Element* fields are made of length elements, each of the width DWords. Each port load cycle handles one data element

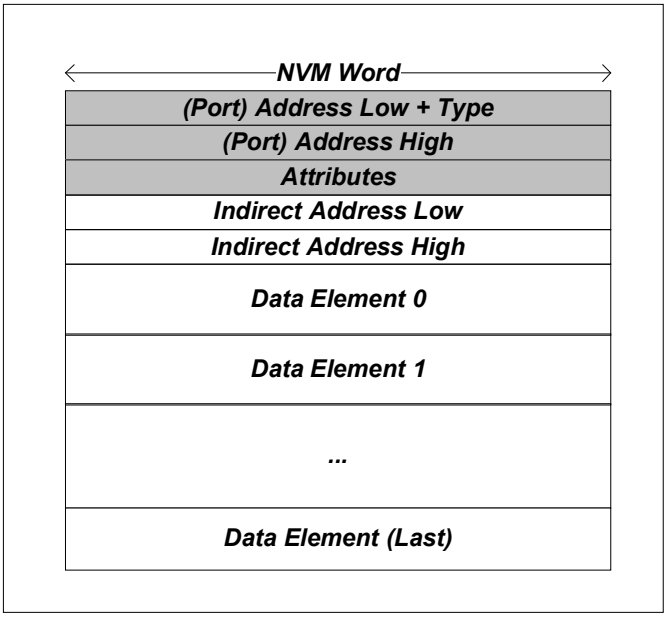


Figure 6-4. Structure of a Type 3 Module

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.

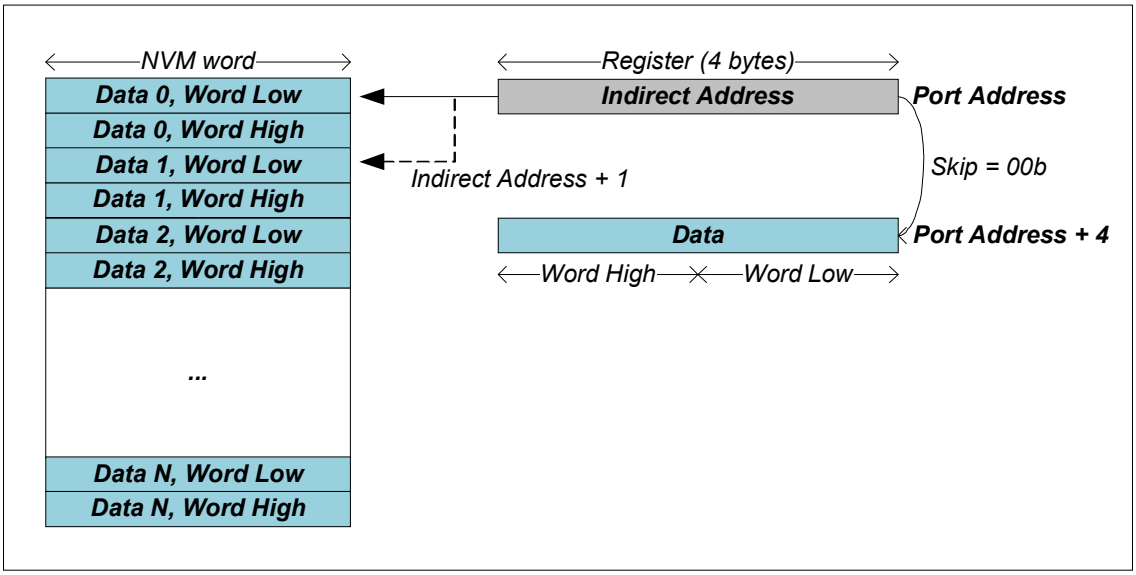


Figure 6-5. Access Through a Type 3 Module of 32-Bit Wide Data Elements

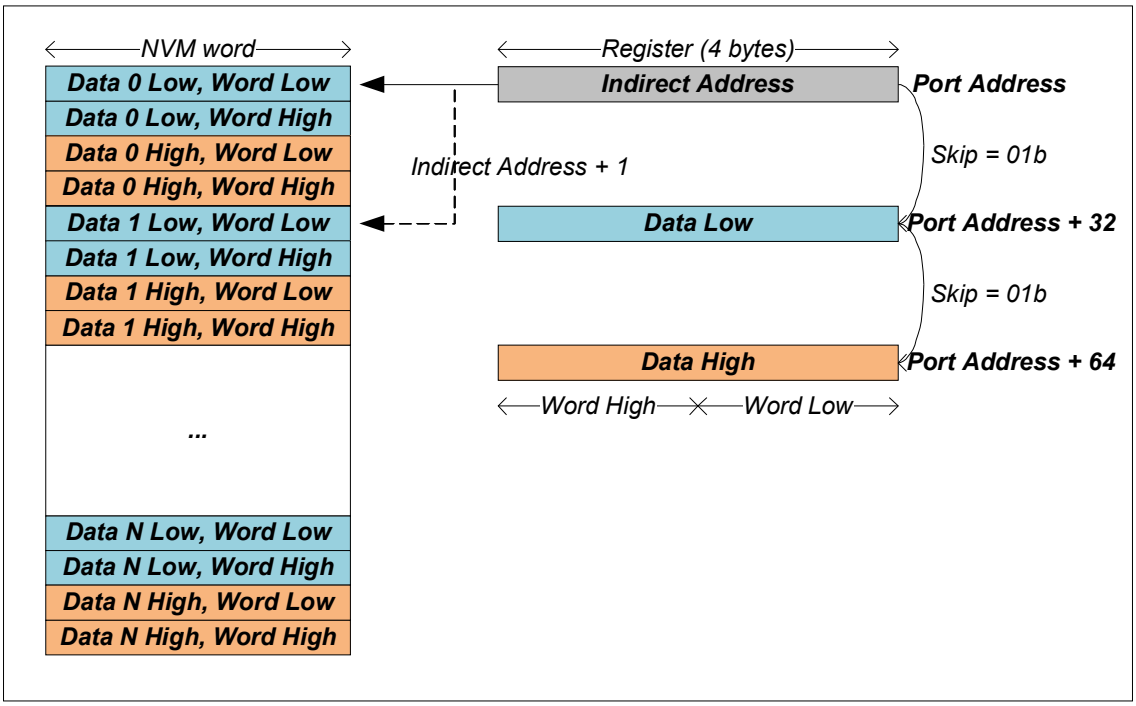


Figure 6-6. Access Through a Type 3 Module of 64-Bit Wide per-Port Data Elements

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).

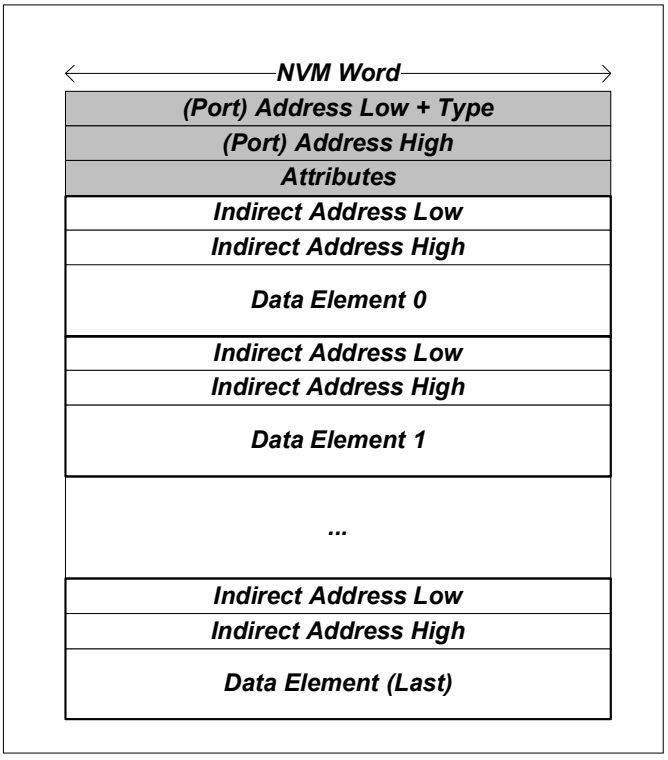


Figure 6-7. Structure of a Type 4 Module

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.

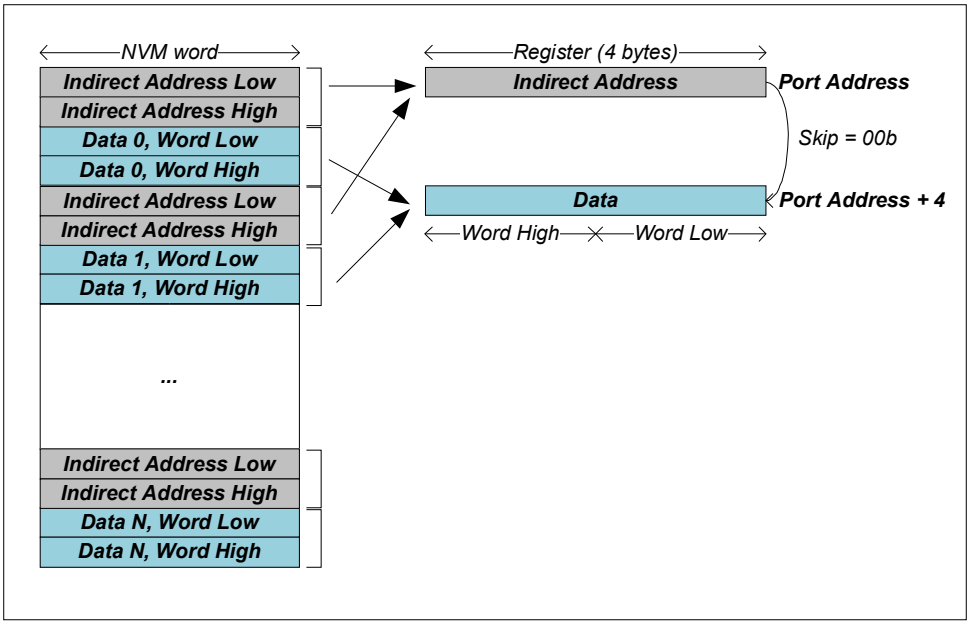


Figure 6-8. Access Through a Type 4 Module of 32-Bit Wide Data Elements

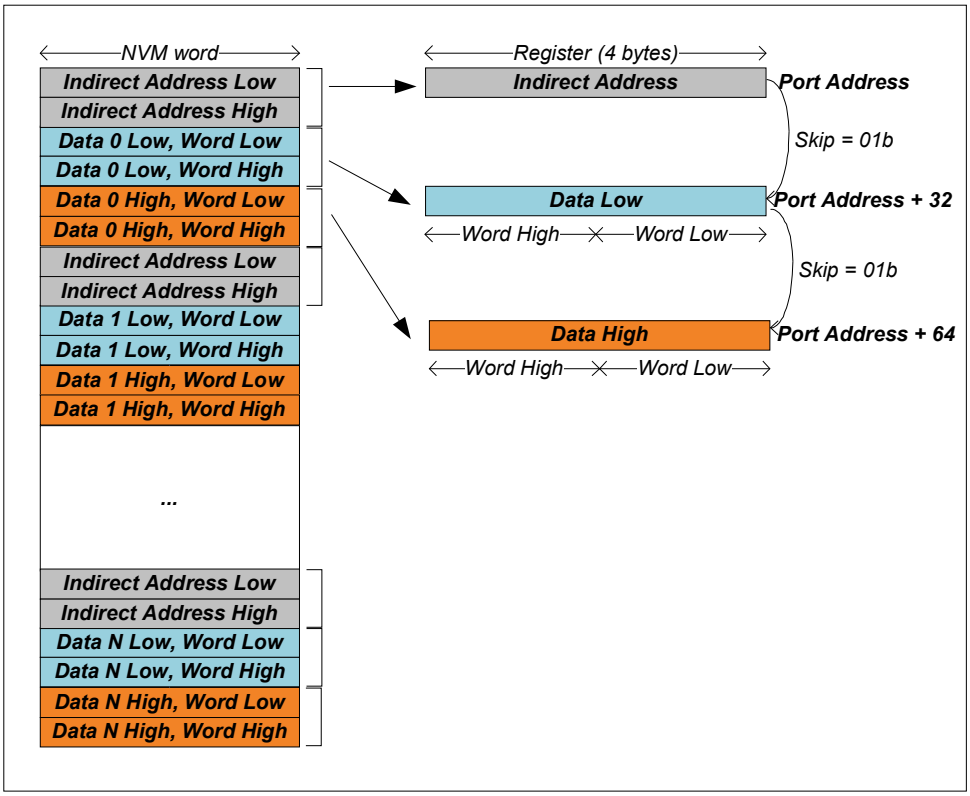


Figure 6-9. Access Through a Type 4 Module of 64-Bit Wide per-Port Data Elements

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 IPs.

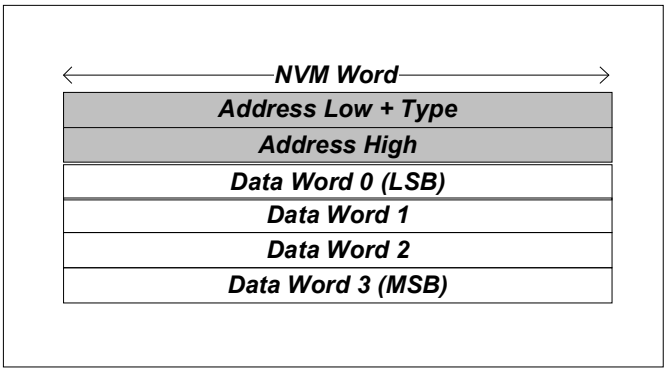


Figure 6-10. Structure of a Type 5 Module

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 auto-load module. Specifically, subtype 0 within Type F module indicates that this module is used to load sideband device. A 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.

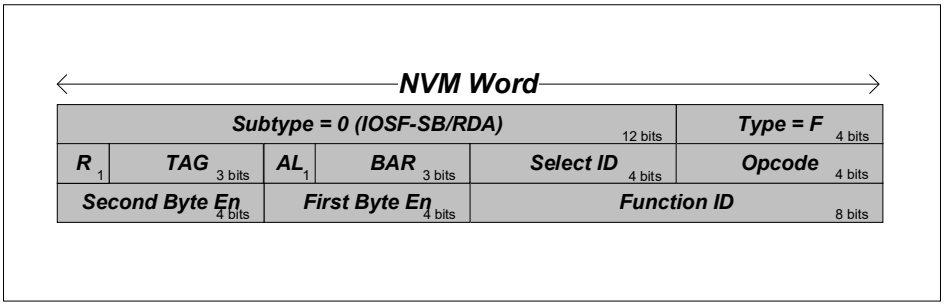


Figure 6-11. Structure of a Type F Module

Two additional 16-bits word 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. 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.
- **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-bit address, the 16 low significant bits are taken from the 28 bits of Type 1, 2, or 5 address. For a 48-bit address, the 28 bits of Type 1, 2, or 5 address are taken and padded with 20 bits of 0.
  - — 0 = 16-bit address
  - — 1 = 48-bit 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 Auto-Load section.

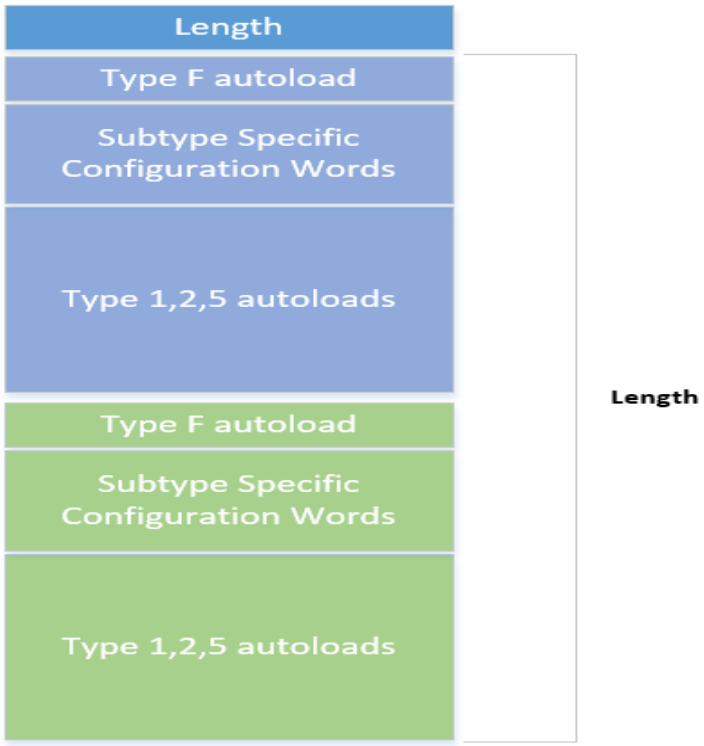


Figure 6-12. Type F Auto-Load Section

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 4 KB. 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.

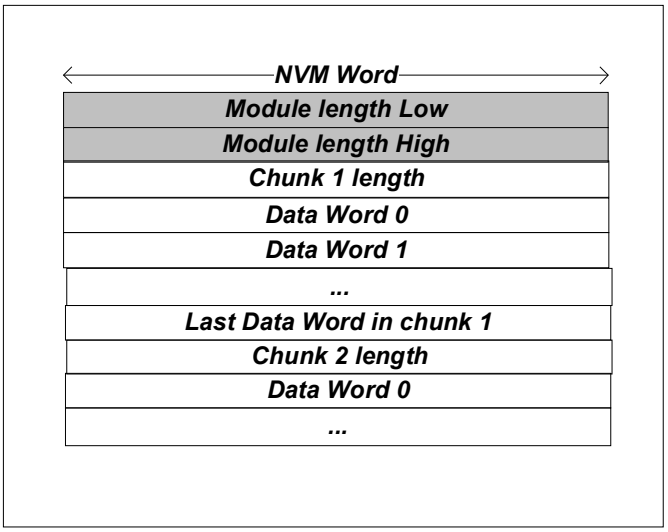


Figure 6-13. Format of Hardware Modules in Flash

6.1.3.8 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 Auto-Load NVM modules according to their loading trigger (see Table 6-2) and with no possibility to perform manual changes.

EMP code and software tools cannot accommodate these variations in mapping offset. Hence, they need a fixed method for accessing some predefined Type 1/2 items. The Auto-Generated Pointers module 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-Generated Pointers module on the detailed NVM map for the list of auto-generated pointers (Section 6.3.60).

6.1.4 NVM Integrity Checks by Software

This section describes the NVM integrity fields inserted in the E810 NVM map to provide a basic detection of a faulty Flash part.

A software checksum 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 the E810.

Modules mapped outside the Shadow RAM are signed, except for the scratch pad areas. EMP is responsible for checking 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 software tools are modifying one of these areas, EMP updates the *Software Checksum* field accordingly.

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 3.4.10.7.

6.1.5 Header of NVM Modules

6.1.5.1 Header of All NVM Modules Mapped to Shadow RAM

Modules read by hardware do not contain pointers to sub-modules and they are not authenticated.

Table 6-3. 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:

- NVM Bank (pointed by NVM word 0x42).
- Option ROM (pointed by NVM word 0x44).

For the NVM Bank, the first 330 words listed in Table 6-4 are mapped at the beginning of the module.

For the Option ROM, the first 330 words listed in Table 6-4 are mapped at the end of the area allocated to the module (at its trailer), though for the sake of the authentication, these words are mapped at the module's header, as listed in Table 6-4.

In Table 6-4, fields colored in cyan are protected by the authentication signature.

Table 6-4. Header of Authenticated NVM Modules

Number of Words	Field or Segment Name	Description and Comments
64	CSS Header	Refer to Section 3.4.9.2.
128	RSA Public Key	Refer to Section 3.4.9. This field is skipped due to SHA256 Hash computing.
2	RSA Exponent	Refer to Section 3.4.9. This field is skipped due to SHA256 Hash computing.
128	Encrypted SHA256 Hash	Refer to Section 3.4.9. This field is skipped due to SHA256 Hash computing.
1	E810 Blank NVM Device ID	A unique, Intel-provided Device ID that identifies the E810 controller among other Intel controllers. It must be set to 0x1590 in 25x25 mm package and 0x1598 in the 21x21 mm package.
2	Max Module Area	It is the maximum Flash area expressed in words that can be used by the module, starting from CSS header (included). It is set to 580K words (1160 KB) for an EMP and PE image modules and to 4K words (8 KB) for other authenticated modules.
2	Current Module Area	It is the Flash area expressed in words that is currently used by the module, starting from CSS header (included).
1	Module Format Version + CRC8	Bit 15 = CRC8 field is used. Set to 1b if a CRC8 is computed over the module, set to 0b otherwise. It is not used and must always be set to 0b. Bits 14:8 = Module format version. Set to 0x02 to use the currently-defined format. Bits 7:0 = CRC8 value.
1	Code Revision	Bits 15:8 = Major revision number. Bits 7:0 = Minor revision number.
1	Reserved Spare Word	Must be zeroed.
N		

6.1.5.3 Module TypeIDs

Each module has a TypeID. The TypeID uniquely identifies the module. The TypeID is used in admin commands to access the module. All modules can be read through the TypeID without knowing the exact location of the module in the NVM. PFA modules that allow direct update can be written using their TypeID.

Table 6-5 list the TypeIDs of all of the modules in the NVM.

Table 6-5. Module TypeID Table

Module TypeID	Section	In PFA	Read Only
0x00	Reserved (Has special meaning in admin commands)		N/A
0x03	GFID Module	No	N/A
0x06	RO PCIR Registers Auto-Load Module		RO
0x07	Auto Generated Pointers Module		RO
0x08	PCIR Registers Auto-Load Module		RO
0x09	EMP Global Module		RO
0x0A	Guarded Zone Module		RO
0x0B	EMP Image Module		RO
0x0C	PE Image Module		RO
0x0E	Manageability Module		RO
0x0F	EMP Settings Module		RO
0x16	PBA Block Module	Yes	RO after manufacturing
0x2F	VPD Module	Yes	RW via VPD access
0x38	POR Registers Auto-Load Module		RO?
0x3B	GLOBR Registers Auto-Load Module		RO
0x3C	CORER Registers Auto-Load Module		RO
0x3D	PHY Configuration Scripts 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 TLV Extension Bank Module		RW
0x48	EMP SR Settings Module		RO
0x49	Feature Configuration Module	Yes	Writable via ANVM only
0x4A	PE CORER Registers Auto-Load Module		RO
0x4B	Link Topology Scratch Pad Area Module		RO
0x4D	Configuration Metadata Module		RO
0x4E	Immediate Values Module	Yes	Writable via ANVM only
0x4F	1588 Parameters Module	Yes	RW
0x50	FW Scratch Pad Area Module		RW
0x53	PE Settings Module		RO

Table 6-5. Module TypeID Table [continued]

Module TypeID	Section	In PFA	Read Only
0x54	PHY Core Post PLL Configuration Module		RO
0x56	External CORER Registers Auto-Load Module		RO
0x58	FW Switch Parameters Module		RO
0x5B	HLP Scratch Pad Area Module		
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	Pass-through Control Words Structure 2 Module	Yes	RW
0x10B	Pass-through Control Words Structure 3 Module	Yes	RW
0x10C	Original EETrack ID Module	Yes	?
0x10F	PF MAC Addresses Module	Yes	RW
0x110	MNG MAC Addresses Module	Yes	RW
0x113	Early PCIR Auto-Load Module (512 bytes)	Yes	RO
0x114	Pass-Through Control Words Structure 4 Module	Yes	RW
0x115	Pass-Through Control Words Structure 5 Module	Yes	RW
0x116	Pass-Through Control Words Structure 6 Module	Yes	RW
0x117	Pass-Through Control Words Structure 7 Module	Yes	RW
0x118	Early POR Auto-Load Module (512 bytes)	Yes	RO
0x119	PSM Preserved Module	Yes	RW
0x11B	Link Topology Netlist Module	No	Updateable via Netlist update flow
0x11D	UART Debug Defaults Module	Yes	RW
0x120	Link Topology Module	Yes	RW
0x127	Component Image Set Version String	Yes	RW
0x128	OEM-Specific Modules	Yes	RW
0x129	LLDP Preserved	Yes	RW
0x130	MinSrev	Yes	RW (with limitations)
0x131	Redfish to the Endpoint (RDE) Parameters	Yes	RW
0x132	CIVD	Yes	RW (as part of NVM update)
0x133	PCIe Serial Number	Yes	RO after manufacturing
0x134	Link Default Override	Yes	RW
0x135	RDMA Control	Yes	RW
0x136	Redfish to the Endpoint (RDE) Parameters - Part II	Yes	RW
0x137	Default DCB Parameters	Yes	RW
0x138	Current DCB Parameters	Yes	RW
0x139	HII Port Disable by Function	Yes	RW
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 follows:

- PFA Header
  - First word of PFA is the Length. Current value is 23 KB in Words (0x2F00).
- 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 6-5 only types that 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.

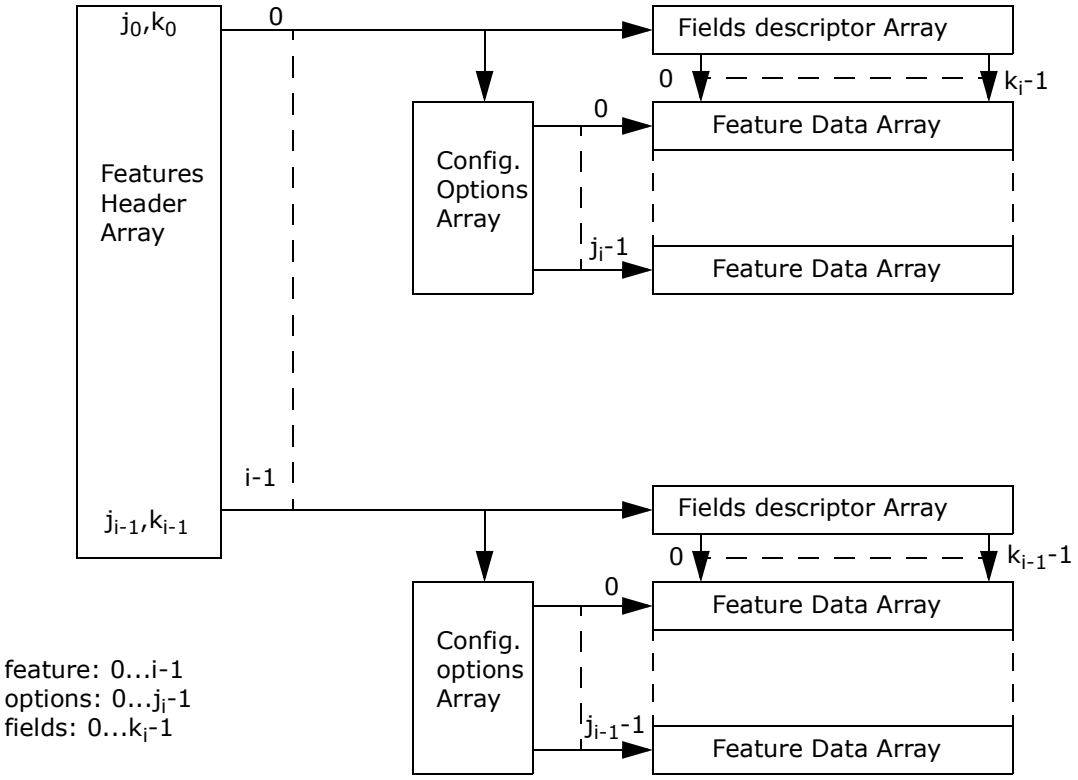


Figure 6-14. Feature Tables Diagram

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 E810 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.

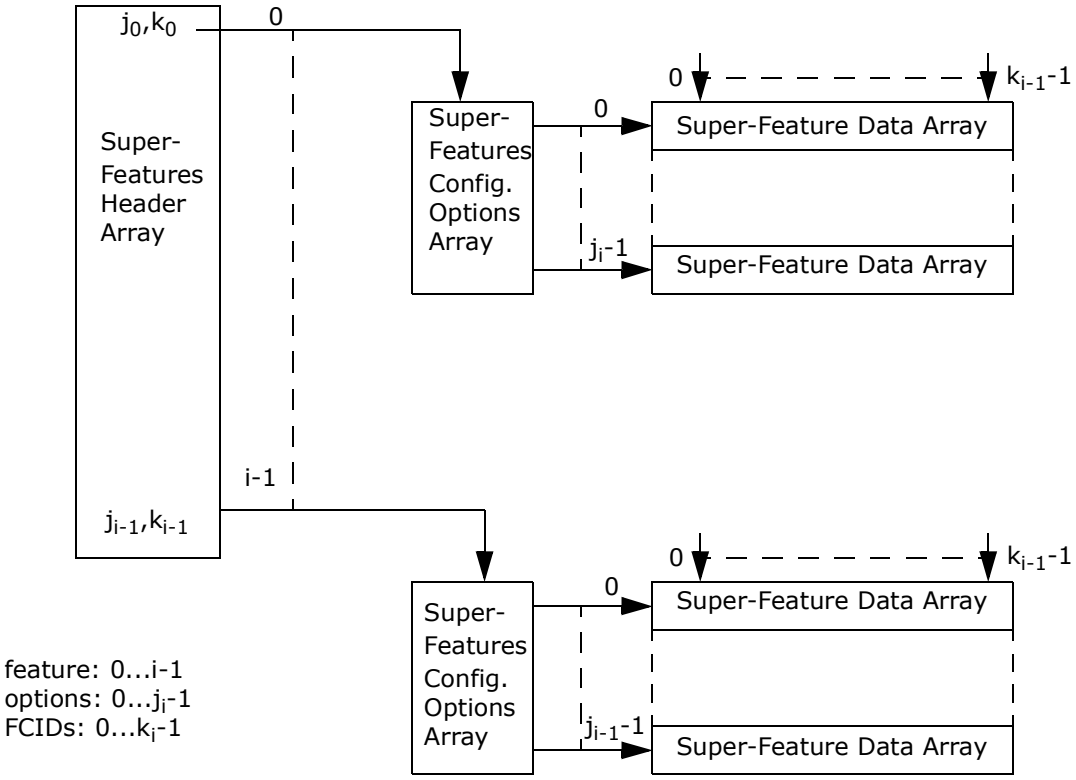


Figure 6-15. Super-Feature Tables Diagram

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 Array1W
...					
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 (ixjxk) 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. 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

This section is part of the PLDM package header. The PLDM header is not part of the NVM image.

Table 6-6. PLDM Header Section Summary Table

Word Address	Used By	Word Name	Section Reference
0x0000	HW	PackageHeaderIdentifier_0	6.2.1.1
0x0001	HW	PackageHeaderIdentifier_1	6.2.1.2
0x0002	HW	PackageHeaderIdentifier_2	6.2.1.3
0x0003	HW	PackageHeaderIdentifier_3	6.2.1.4
0x0004	HW	PackageHeaderIdentifier_4	6.2.1.5
0x0005	HW	PackageHeaderIdentifier_5	6.2.1.6
0x0006	HW	PackageHeaderIdentifier_6	6.2.1.7
0x0007	HW	PackageHeaderIdentifier_7	6.2.1.8
0x0008	HW	FormatRevision_HeaderSize_LSB	6.2.1.9
0x0009	HW	Headersize_MSB	6.2.1.10
0x000A	HW	ReleaseDateTime_0	6.2.1.11
0x000B	HW	ReleaseDateTime_1	6.2.1.12
0x000C	HW	ReleaseDateTime_2	6.2.1.13
0x000D	HW	ReleaseDateTime_3	6.2.1.14
0x000E	HW	ReleaseDateTime_4	6.2.1.15
0x000F	HW	ReleaseDateTime_5	6.2.1.16
0x0010	HW	ComponentBitmapBitLength	6.2.1.17
0x0011	HW	PackageVersionStringType_Length	6.2.1.18
0x0012	HW	PackageVersionString- FW Component Prefix	6.2.1.19
0x0013	HW	PackageVersionString - FW Component Version 0	6.2.1.20
0x0014	HW	PackageVersionString - FW Component Version 1	6.2.1.21
0x0015	HW	PackageVersionString - FW Component Version 2	6.2.1.22
0x0016	HW	PackageVersionString - FW Component Version 3	6.2.1.23
0x0017	HW	PackageVersionString - OROM Component Prefix	6.2.1.24
0x0018	HW	PackageVersionString - OROM Component Version 0	6.2.1.25
0x0019	HW	PackageVersionString - OROM Component Version 1	6.2.1.26
0x001A	HW	PackageVersionString - OROM Component Version 2	6.2.1.27
0x001B	HW	PackageVersionString - OROM Component Version 3	6.2.1.28
0x001C	HW	PackageVersionString - Netlist Component Prefix	6.2.1.29
0x001D	HW	PackageVersionString - Netlist Component Version 0	6.2.1.30
0x001E	HW	PackageVersionString - Netlist Component Version 1	6.2.1.31

Table 6-6. PLDM Header Section Summary Table [continued]

Word Address	Used By	Word Name	Section Reference
0x001F	HW	PackageVersionString - Netlist Component Version 2	6.2.1.32
0x0020	HW	PackageVersionString - Netlist Component Version 3	6.2.1.33
0x0021	HW	DeviceIDRecordCount and Last Byte of PackageVersionString	6.2.1.34
0x0022	HW	Recordlength	6.2.1.35
0x0023	HW	DescriptorCount and DeviceUpdateOptionFlags LSB	6.2.1.36
0x0024	HW	DeviceUpdateOptionFlags - Middle	6.2.1.37
0x0025	HW	DeviceUpdateOptionFlags - MSB and String Type	6.2.1.38
0x0026	HW	ComponentImageSetVersionStringLength and FirmwareDevicePackageDataLength - LSB	6.2.1.39
0x0027	HW	FirmwareDevicePackageDataLength - MSB and ApplicableComponents	6.2.1.40
0x0028	HW	ComponentImageSetVersionString- FW Component Prefix	6.2.1.41
0x0029	HW	ComponentImageSetVersionString - FW Component Version 0	6.2.1.42
0x002A	HW	ComponentImageSetVersionString - FW Component Version 1	6.2.1.43
0x002B	HW	ComponentImageSetVersionString - FW Component Version 2	6.2.1.44
0x002C	HW	ComponentImageSetVersionString - FW Component Version 3	6.2.1.45
0x002D	HW	ComponentImageSetVersionString - OROM Component Prefix	6.2.1.46
0x002E	HW	ComponentImageSetVersionString - OROM Component Version 0	6.2.1.47
0x002F	HW	ComponentImageSetVersionString - OROM Component Version 1	6.2.1.48
0x0030	HW	ComponentImageSetVersionString - OROM Component Version 2	6.2.1.49
0x0031	HW	ComponentImageSetVersionString - OROM Component Version 3	6.2.1.50
0x0032	HW	ComponentImageSetVersionString - Netlist Component Prefix	6.2.1.51
0x0033	HW	ComponentImageSetVersionString - Netlist Component Version 0	6.2.1.52
0x0034	HW	ComponentImageSetVersionString - Netlist Component Version 1	6.2.1.53
0x0035	HW	ComponentImageSetVersionString - Netlist Component Version 2	6.2.1.54
0x0036	HW	ComponentImageSetVersionString - Netlist Component Version 3	6.2.1.55
0x0037	HW	ComponentImageSetVersionString - Null Padding	6.2.1.56
0x0038	HW	InitialDescriptorType	6.2.1.57
0x0039	HW	InitialDescriptorLength	6.2.1.58
0x003A	HW	InitialDescriptorData	6.2.1.59
0x003B	HW	AdditionalDescriptorType - DeviceID	6.2.1.60
0x003C	HW	AdditionalDescriptorLength - DeviceID	6.2.1.61
0x003D	HW	AdditionalDescriptorData - Device ID	6.2.1.62
0x003E	HW	AdditionalDescriptorType - SubVendorID	6.2.1.63
0x003F	HW	AdditionalDescriptorLength - SubVendorID	6.2.1.64
0x0040	HW	AdditionalDescriptorIdentifierData - SubVendorID	6.2.1.65
0x0041	HW	AdditionalDescriptorType - SubSystemD	6.2.1.66
0x0042	HW	AdditionalDescriptorLength - SubSystemD	6.2.1.67
0x0043	HW	AdditionalDescriptorIdentifierData - SubSystemD	6.2.1.68

Table 6-6. PLDM Header Section Summary Table [continued]

Word Address	Used By	Word Name	Section Reference
0x0044	HW	FirmwareDevicePackageData - Header	6.2.1.69
0x0045	HW	FirmwareDevicePackageData - GFID TLV Type	6.2.1.70
0x0046	HW	FirmwareDevicePackageData - GFID TLV Length	6.2.1.71
0x0047	HW	FirmwareDevicePackageData - GFID TLV Value - Current GFID IANA	6.2.1.72
0x0048	HW	FirmwareDevicePackageData - GFID TLV Value - Current GFID DeviceID	6.2.1.73
0x0049 + 1*n, n=0...15	HW	FirmwareDevicePackageData - GFID TLV Value - Current GFID Zeros	6.2.1.74
0x0059	HW	FirmwareDevicePackageData - GFID TLV Value - Original GFID IANA	6.2.1.75
0x005A	HW	FirmwareDevicePackageData - GFID TLV Value - Original GFID DeviceID	6.2.1.76
0x005B + 1*n, n=0...15	HW	FirmwareDevicePackageData - GFID TLV Value - Original GFID Zeros	6.2.1.77
0x006B	HW	FirmwareDevicePackageData - Additional TLVs	6.2.1.78
0x006C	HW	ComponentImageCount	6.2.1.79
0x006D	HW	ComponentClassification	6.2.1.80
0x006E	HW	ComponentIdentifier	6.2.1.81
0x006F	HW	ComponentComparisonStamp LSB	6.2.1.82
0x0070	HW	ComponentComparisonStamp MSB	6.2.1.83
0x0071	HW	ComponentOptions	6.2.1.84
0x0072	HW	RequestedComponentActivationMethod	6.2.1.85
0x0073	HW	ComponentLocationOffset LSB	6.2.1.86
0x0074	HW	ComponentLocationOffset MSB	6.2.1.87
0x0075	HW	ComponentSize LSB	6.2.1.88
0x0076	HW	ComponentSize MSB	6.2.1.89
0x0077	HW	ComponentVersionStringTypeAndLength	6.2.1.90
0x0078	HW	ComponentVersionString- Dev Starter Major	6.2.1.91
0x0079	HW	ComponentVersionString - Dev Starter Minor	6.2.1.92
0x007A	HW	ComponentVersionString - EETRACK-ID MSB	6.2.1.93
0x007B	HW	ComponentVersionString - EETRACK-ID LSB	6.2.1.94
0x007C	HW	ComponentVersionString - Dot and Srev Byte 7	6.2.1.95
0x007D	HW	ComponentVersionString - Srev Bytes 6-5	6.2.1.96
0x007E	HW	ComponentVersionString - Srev Bytes 4-3	6.2.1.97
0x007F	HW	ComponentVersionString - Srev Bytes 2-1	6.2.1.98
0x0080	HW	ComponentVersionString - Srev Byte 0 and Null	6.2.1.99
0x0081	HW	ComponentClassification	6.2.1.100
0x0082	HW	ComponentIdentifier	6.2.1.101
0x0083	HW	ComponentComparisonStamp LSB	6.2.1.102
0x0084	HW	ComponentComparisonStamp MSB	6.2.1.103
0x0085	HW	ComponentOptions	6.2.1.104
0x0086	HW	RequestedComponentActivationMethod	6.2.1.105

Table 6-6. PLDM Header Section Summary Table [continued]

Word Address	Used By	Word Name	Section Reference
0x0087	HW	ComponentLocationOffset LSB	6.2.1.106
0x0088	HW	ComponentLocationOffset MSB	6.2.1.107
0x0089	HW	ComponentSize LSB	6.2.1.108
0x008A	HW	ComponentSize MSB	6.2.1.109
0x008B	HW	ComponentVersionStringTypeAndLength	6.2.1.110
0x008C	HW	ComponentVersionString- CIVD High MSB	6.2.1.111
0x008D	HW	ComponentVersionString - CIVD High LSB	6.2.1.112
0x008E	HW	ComponentVersionString - CIVD Low MSB	6.2.1.113
0x008F	HW	ComponentVersionString - CIVD Low LSB	6.2.1.114
0x0090	HW	ComponentVersionString - Dot and Srev Byte 7	6.2.1.115
0x0091	HW	ComponentVersionString - Srev Bytes 6-5	6.2.1.116
0x0092	HW	ComponentVersionString - Srev Bytes 4-3	6.2.1.117
0x0093	HW	ComponentVersionString - Srev Bytes 2-1	6.2.1.118
0x0094	HW	ComponentVersionString - Srev Byte 0 and Null	6.2.1.119
0x0095	HW	ComponentClassification	6.2.1.120
0x0096	HW	ComponentIdentifier	6.2.1.121
0x0097	HW	ComponentComparisonStamp LSB	6.2.1.122
0x0098	HW	ComponentComparisonStamp MSB	6.2.1.123
0x0099	HW	ComponentOptions	6.2.1.124
0x009A	HW	RequestedComponentActivationMethod	6.2.1.125
0x009B	HW	ComponentLocationOffset LSB	6.2.1.126
0x009C	HW	ComponentLocationOffset MSB	6.2.1.127
0x009D	HW	ComponentSize LSB	6.2.1.128
0x009E	HW	ComponentSize MSB	6.2.1.129
0x009F	HW	ComponentVersionStringTypeAndLength	6.2.1.130
0x00A0	HW	ComponentVersionString- ReleaseVersion Major Bytes 7-6	6.2.1.131
0x00A1	HW	ComponentVersionString - ReleaseVersion Major Bytes 5-4	6.2.1.132
0x00A2	HW	ComponentVersionString - ReleaseVersion Major Bytes 3-2	6.2.1.133
0x00A3	HW	ComponentVersionString - ReleaseVersion Major Bytes 1-0	6.2.1.134
0x00A4	HW	ComponentVersionString - Dot and ReleaseVersion Minor Byte 7	6.2.1.135
0x00A5	HW	ComponentVersionString - ReleaseVersion Minor Bytes 6-5	6.2.1.136
0x00A6	HW	ComponentVersionString - ReleaseVersion Minor Bytes 4-3	6.2.1.137
0x00A7	HW	ComponentVersionString - ReleaseVersion Minor Bytes 2-1	6.2.1.138
0x00A8	HW	ComponentVersionString - ReleaseVersion Minor Byte 0 and Dot	6.2.1.139
0x00A9	HW	ComponentVersionString - ReleaseVersion Type Bytes 7-6	6.2.1.140
0x00AA	HW	ComponentVersionString - ReleaseVersion Type Bytes 5-4	6.2.1.141
0x00AB	HW	ComponentVersionString - ReleaseVersion Type Bytes 3-2	6.2.1.142

Table 6-6. PLDM Header Section Summary Table [continued]

Word Address	Used By	Word Name	Section Reference
0x00AC	HW	ComponentVersionString - ReleaseVersion Type Bytes 0-1	6.2.1.143
0x00AD	HW	ComponentVersionString - Dot and Customer Netlist IANA Byte 7	6.2.1.144
0x00AE	HW	ComponentVersionString - Customer Netlist IANA Bytes 6-5	6.2.1.145
0x00AF	HW	ComponentVersionString - Customer Netlist IANA Bytes 4-3	6.2.1.146
0x00B0	HW	ComponentVersionString - Customer Netlist IANA Bytes 2-1	6.2.1.147
0x00B1	HW	ComponentVersionString - Customer Netlist IANA Byte 0 and Dot	6.2.1.148
0x00B2	HW	ComponentVersionString - Customer Netlist Version Bytes 3-2	6.2.1.149
0x00B3	HW	ComponentVersionString - Customer Netlist Version Bytes 1-0	6.2.1.150
0x00B4	HW	ComponentVersionString - Nulls	6.2.1.151
0x00B5	HW	PackageHeaderChecksum - LSB	6.2.1.152
0x00B6	HW	PackageHeaderChecksum - MSB	6.2.1.153

6.2.1.1 PackageHeaderIdentifier_0 (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PackageHeaderIdentifier	0xCA02	

6.2.1.2 PackageHeaderIdentifier_1 (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PackageHeaderIdentifier	0x059A	

6.2.1.3 PackageHeaderIdentifier_2 (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PackageHeaderIdentifier	0xA02F	

6.2.1.4 PackageHeaderIdentifier_3 (0x0003)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PackageHeaderIdentifier	0x9800	

6.2.1.5 PackageHeaderIdentifier_4 (0x0004)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PackageHeaderIdentifier	0x4943	

6.2.1.6 PackageHeaderIdentifier_5 (0x0005)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PackageHeaderIdentifier	0xCB7D	

6.2.1.7 PackageHeaderIdentifier_6 (0x0006)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PackageHeaderIdentifier	0x878C	

6.2.1.8 PackageHeaderIdentifier_7 (0x0007)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PackageHeaderIdentifier	0xF018	

6.2.1.9 FormatRevision_HeaderSize_LSB (0x0008)

Bit(s)	Field Name	Default NVM Value	Description
15:8	HeaderSize_LSB		
7:0	FormatRevision	0x01	

6.2.1.10 Headersize_MSB (0x0009)

Bit(s)	Field Name	Default NVM Value	Description
15:8	RelelaseDateTimeUTC_offset_LSB	0x0	timestamp104 byte 0.
7:0	Headersize_MSB		

6.2.1.11 ReleaseDateTime_0 (0x000A)

Bit(s)	Field Name	Default NVM Value	Description
15:8	RelelaseDateTimeMsec_0	0x0	timestamp104 byte 2.
7:0	RelelaseDateTimeUTC_offset_MSB_by	0x0	timestamp104 byte 1.

6.2.1.12 ReleaseDateTime_1 (0x000B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	RelelaseDateTimeMsec_1_2	0x0	timestamp104 byte 3:4.

6.2.1.13 ReleaseDateTime_2 (0x000C)

Bit(s)	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)

Bit(s)	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)

Bit(s)	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)

Bit(s)	Field Name	Default NVM Value	Description
15:12	ReleaseDateTime UTC Resolution	0x2	timestamp104 byte 12 [7:4]. Resolution is minutes.
11:8	ReleaseDateTime Resolution	0x7	timestamp104 byte 12 [3:0]. Resolution is minutes.
7:0	ReleaseDateTimeYear_MSB	0x0	timestamp104 byte 11.

6.2.1.17 ComponentBitmapBitLength (0x0010)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentBitmapBitLength	0x8	

6.2.1.18 PackageVersionStringType_Length (0x0011)

Bit(s)	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)

Bit(s)	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)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PackageVersionString - FW Component Version 0	0x3030	Firmware component version 0. ASCII presentation of firmware comparison stamp. Values are zero (in ASCII) padded.

6.2.1.21 PackageVersionString - FW Component Version 1 (0x0014)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PackageVersionString - FW Component Version 1	0x3030	Firmware component version 1. ASCII presentation of firmware comparison stamp. Values are zero (in ASCII) padded.

6.2.1.22 PackageVersionString - FW Component Version 2 (0x0015)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PackageVersionString - FW Component Version 2	0x3030	Firmware component version 2. ASCII presentation of firmware comparison stamp. Values are zero (in ASCII) padded.

6.2.1.23 PackageVersionString - FW Component Version 3 (0x0016)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PackageVersionString - FW Component Version 3	0x3030	Firmware component version 3. ASCII presentation of firmware comparison stamp. Values are zero (in ASCII) padded.

6.2.1.24 PackageVersionString - OROM Component Prefix (0x0017)

Bit(s)	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)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PackageVersionString - OROM Component Version 0	0x3030	OROM component version 0. ASCII presentation of OROM comparison stamp. Values are zero (in ASCII) padded.

6.2.1.26 PackageVersionString - OROM Component Version 1 (0x0019)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PackageVersionString - OROM Component Version 1	0x3030	OROM component version 1. ASCII presentation of OROM comparison stamp. Values are zero (in ASCII) padded.

6.2.1.27 PackageVersionString - OROM Component Version 2 (0x001A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PackageVersionString - OROM Component Version 2	0x3030	OROM component version 2. ASCII presentation of OROM comparison stamp. Values are zero (in ASCII) padded.

6.2.1.28 PackageVersionString - OROM Component Version 3 (0x001B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PackageVersionString - OROM Component Version 3	0x3030	OROM component version 3. ASCII presentation of OROM comparison stamp. Values are zero (in ASCII) padded.

6.2.1.29 PackageVersionString - Netlist Component Prefix (0x001C)

Bit(s)	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)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PackageVersionString - Netlist Component Version 0	0x3030	Netlist component version 0. ASCII presentation of Netlist comparison stamp. Values are zero (in ASCII) padded.

6.2.1.31 PackageVersionString - Netlist Component Version 1 (0x001E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PackageVersionString - Netlist Component Version 1	0x3030	Netlist component version 1. ASCII presentation of Netlist comparison stamp. Values are zero (in ASCII) padded.

6.2.1.32 PackageVersionString - Netlist Component Version 2 (0x001F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PackageVersionString - Netlist Component Version 2	0x3030	Netlist component version 2. ASCII presentation of Netlist comparison stamp. Values are zero (in ASCII) padded.

6.2.1.33 PackageVersionString - Netlist Component Version 3 (0x0020)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PackageVersionString - Netlist Component Version 3	0x3030	Netlist component version 3. ASCII presentation of Netlist comparison stamp. Values are zero (in ASCII) padded.

6.2.1.34 DeviceIDRecordCount and Last Byte of PackageVersionString (0x0021)

Bit(s)	Field Name	Default NVM Value	Description
15:8	DeviceIDRecordCount	0x1	
7:0	PackageVersionString Last Byte	0x0	

6.2.1.35 Recordlength (0x0022)

Bit(s)	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 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)

Bit(s)	Field Name	Default NVM Value	Description
15:9	DeviceUpdateOptionFlags LSB	0x0	Bits [31:24] reserved.
8	DeviceUpdateOptionFlags LSB - Continue Component Updates After Failure	0b	Bits [31:24] reserved.
7:0	DescriptorCount	0x1	The number of descriptors included within the *RecordDescriptors* field for this record.

6.2.1.37 DeviceUpdateOptionFlags - Middle (0x0024)

Bit(s)	Field Name	Default NVM Value	Description
15:0	DeviceUpdateOptionFlaDeviceUpdateOptionFlags - Middlegs -Middle	0x0	Bits [23:8] reserved.

6.2.1.38 DeviceUpdateOptionFlags - MSB and String Type (0x0025)

Bit(s)	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)

Bit(s)	Field Name	Default NVM Value	Description
15:8	FirmwareDevicePackageDataLength - LSB		
7:0	ComponentImageSetVersionStringLength	0x20	

6.2.1.40 FirmwareDevicePackageDataLength - MSB and ApplicableComponents (0x0027)

Bit(s)	Field Name	Default NVM Value	Description
15:11	Reserved ApplicableComponents	0x0	
10	FW Link topology ApplicableComponent	0b	0b = Non Applicable 1b = Applicable
9	OROM ApplicableComponent	1b	0b = Non Applicable 1b = Applicable
8	NVM ApplicableComponent	1b	0b = Non Applicable 1b = Applicable
7:0	FirmwareDevicePackageDataLength - MSB		

6.2.1.41 ComponentImageSetVersionString- FW Component Prefix (0x0028)

Bit(s)	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)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - FW Component Version 0	0x3030	Firmware component version 0. ASCII presentation of firmware comparison stamp. Values are zero (in ASCII) padded.

6.2.1.43 ComponentImageSetVersionString - FW Component Version 1 (0x002A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - FW Component Version 1	0x3030	Firmware component version 1. ASCII presentation of firmware comparison stamp. Values are zero (in ASCII) padded.

6.2.1.44 ComponentImageSetVersionString - FW Component Version 2 (0x002B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - FW Component Version 2	0x3030	Firmware component version 2. ASCII presentation of firmware comparison stamp. Values are zero (in ASCII) padded.

6.2.1.45 ComponentImageSetVersionString - FW Component Version 3 (0x002C)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - FW Component Version 3	0x3030	Firmware component version 3. ASCII presentation of firmware comparison stamp. Values are zero (in ASCII) padded.

6.2.1.46 ComponentImageSetVersionString - OROM Component Prefix (0x002D)

Bit(s)	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)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - OROM Component Version 0	0x3030	OROM component version 0. ASCII presentation of OROM comparison stamp. Values are zero (in ASCII) padded.

6.2.1.48 ComponentImageSetVersionString - OROM Component Version 1 (0x002F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - OROM Component Version 1	0x3030	OROM component version 1. ASCII presentation of OROM comparison stamp. Values are zero (in ASCII) padded.

6.2.1.49 ComponentImageSetVersionString - OROM Component Version 2 (0x0030)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - OROM Component Version 2	0x3030	OROM component version 2. ASCII presentation of OROM comparison stamp. Values are zero (in ASCII) padded.

6.2.1.50 ComponentImageSetVersionString - OROM Component Version 3 (0x0031)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - OROM Component Version 3	0x3030	OROM component version 3. ASCII presentation of OROM comparison stamp. Values are zero (in ASCII) padded.

6.2.1.51 ComponentImageSetVersionString - Netlist Component Prefix (0x0032)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSet VersionString - Netlist Component Prefix	0x3A54	"T:" in ASCII.

6.2.1.52 ComponentImageSetVersionString - Netlist Component Version 0 (0x0033)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - Netlist Component Version 0	0x3030	Netlist component version 0. ASCII presentation of Netlist comparison stamp. Values are zero (in ASCII) padded.

6.2.1.53 ComponentImageSetVersionString - Netlist Component Version 1 (0x0034)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - Netlist Component Version 1	0x3030	Netlist component version 1. ASCII presentation of Netlist comparison stamp. Values are zero (in ASCII) padded.

6.2.1.54 ComponentImageSetVersionString - Netlist Component Version 2 (0x0035)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - Netlist Component Version 2	0x3030	Netlist component version 2. ASCII presentation of Netlist comparison stamp. Values are zero (in ASCII) padded.

6.2.1.55 ComponentImageSetVersionString - Netlist Component Version 3 (0x0036)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - Netlist Component Version 3	0x3030	Netlist component version 3. ASCII presentation of Netlist comparison stamp. Values are zero (in ASCII) padded.

6.2.1.56 ComponentImageSetVersionString - Null Padding (0x0037)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - Null Padding	0x0	

6.2.1.57 InitialDescriptorType (0x0038)

Bit(s)	Field Name	Default NVM Value	Description
15:0	InitialDescriptorType	0x0	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). 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)

Bit(s)	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)

Bit(s)	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)

Bit(s)	Field Name	Default NVM Value	Description
15:0	AdditionalDescriptorType	0x100	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). 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)

Bit(s)	Field Name	Default NVM Value	Description
15:0	AdditionalDescriptorLength	0x2	Indicates the length, in bytes, of the *InitialDescriptorData* field.

6.2.1.62 AdditionalDescriptorData - Device ID (0x003D)

Bit(s)	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)

Bit(s)	Field Name	Default NVM Value	Description
15:0	AdditonialDescriptorType	0x101	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). 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)

Bit(s)	Field Name	Default NVM Value	Description
15:0	AdditonialDescriptorLength	0x2	Indicates the length, in bytes, of the *InitialDescriptorData* field.

6.2.1.65 AdditionalDescriptorIdentifierData - SubVendorID (0x0040)

Bit(s)	Field Name	Default NVM Value	Description
15:0	SubVendorID	0x8086	PCI Vendor ID assigned to the FD vendor.

6.2.1.66 AdditionalDescriptorType - SubSystemD (0x0041)

Bit(s)	Field Name	Default NVM Value	Description
15:0	AdditonialDescriptorType	0x102	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). 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)

Bit(s)	Field Name	Default NVM Value	Description
15:0	AdditonialDescriptorLength	0x2	Indicates the length, in bytes, of the *InitialDescriptorData* field.

6.2.1.68 AdditionalDescriptorIdentifierData - SubSystemD (0x0043)

Bit(s)	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 that the UA transfers to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and simply passes the contents of this field when the FD requests it via the *GetPackageData* command response.

If the *FirmwareDevicePackageDataLength* field is set to 0x0000, this field contains no data and is zero bytes in length.

Bit(s)	Field Name	Default NVM Value	Description
15:2	Reserved	0x0	Reserved.
1	O-ROM Version to be Set in VPD	0x0	0b = Do not set O-ROM version in VPD. 1b = 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 that the UA transfers to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and simply passes the contents of this field when the FD requests it via the *GetPackageData* command response.

If the *FirmwareDevicePackageDataLength* field is set to 0x0000, this field contains no data and is zero bytes in length.

Bit(s)	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 that the UA transfers to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and simply passes the contents of this field when the FD requests it via the *GetPackageData* command response.

If the *FirmwareDevicePackageDataLength* field is set to 0x0000, this field contains no data and is zero bytes in length.

Bit(s)	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 that the UA transfers to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and simply passes the contents of this field when the FD requests it via the *GetPackageData* command response.

If the *FirmwareDevicePackageDataLength* field is set to 0x0000, this field contains no data and is zero bytes in length.

Bit(s)	Field Name	Default NVM Value	Description
15:0	TLV Value-IANA	0xB85C	

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 that the UA transfers to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and simply passes the contents of this field when the FD requests it via the *GetPackageData* command response.

If the *FirmwareDevicePackageDataLength* field is set to 0x0000, this field contains no data and is zero bytes in length.

Bit(s)	Field Name	Default NVM Value	Description
15:0	TLV Value - device ID	0x1590	Valid values are: 0x1590 = E810 25x25 mm package (E810-CAM2/CAM1) 0x1598 = E810 21x21 mm package (E810-XXVAM2)

6.2.1.74 FirmwareDevicePackageData - GFID TLV Value - Current GFID Zeros[n] (0x0049 + 1*n, n=0...15)

An optional data field that can be provided within the firmware update package that the UA transfers to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and simply passes the contents of this field when the FD requests it via the *GetPackageData* command response.

If the *FirmwareDevicePackageDataLength* field is set to 0x0000, this field contains no data and is zero bytes in length.

Bit(s)	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.75 FirmwareDevicePackageData - GFID TLV Value - Original GFID IANA (0x0059)

An optional data field that can be provided within the firmware update package that the UA transfers to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and simply passes the contents of this field when the FD requests it via the *GetPackageData* command response.

If the *FirmwareDevicePackageDataLength* field is set to 0x0000, this field contains no data and is zero bytes in length.

Bit(s)	Field Name	Default NVM Value	Description
15:0	TLV Value-IANA	0xB85C	

6.2.1.76 FirmwareDevicePackageData - GFID TLV Value - Original GFID DeviceID (0x005A)

An optional data field that can be provided within the firmware update package that the UA transfers to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and simply passes the contents of this field when the FD requests it via the *GetPackageData* command response.

If the *FirmwareDevicePackageDataLength* field is set to 0x0000, this field contains no data and is zero bytes in length.

Bit(s)	Field Name	Default NVM Value	Description
15:0	TLV Value - Device ID	0x1590	Valid values are: 0x1590 = E810 25x25 mm package (E810-CAM2/CAM1) 0x1598 = E810 21x21 mm package (E810-XXVAM2)

6.2.1.77 FirmwareDevicePackageData - GFID TLV Value - Original GFID Zeros[n] (0x005B + 1*n, n=0...15)

An optional data field that can be provided within the firmware update package that the UA transfers to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and simply passes the contents of this field when the FD requests it via the *GetPackageData* command response.

If the *FirmwareDevicePackageDataLength* field is set to 0x0000, this field contains no data and is zero bytes in length.

Bit(s)	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.78 FirmwareDevicePackageData - Additional TLVs (0x006B)

Raw data module length: variable

TLVs needed for firmware update over PLDM.

6.2.1.79 ComponentImageCount (0x006C)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageCount	0x3	

6.2.1.80 ComponentClassification (0x006D)

Bit(s)	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

6.2.1.81 ComponentIdentifier (0x006E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentIdentifier	0x6	Valid values are: 0x1 = TLV Extension 0x2 = Link Topology Scratch Pad Area 0x3 = Descriptor Block 0x5 = OROM 0x6 = NVM 0x8 = Topology Netlist

6.2.1.82 ComponentComparisonStamp LSB (0x006F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentComparisonStamp Low	0x0	

6.2.1.83 ComponentComparisonStamp MSB (0x0070)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentComparisonStamp High	0x0	

6.2.1.84 ComponentOptions (0x0071)

Bit(s)	Field Name	Default NVM Value	Description
15:2	Reserved	0x0	Reserved.
1	Use Component Comparison Stamp	1b	0b = Disabled 1b = Enabled
0	Force Update	1b	0b = Disabled 1b = Enabled

6.2.1.85 RequestedComponentActivationMethod (0x0072)

Bit(s)	Field Name	Default NVM Value	Description
15:6	Reserved		Reserved.
5	AC Power Cycle	0b	0b = Disabled 1b = Enabled
4	DC Power Cycle	0b	0b = Disabled 1b = Enabled
3	System Reboot	1b	0b = Disabled 1b = Enabled
2	Medium-Specific Reset	0b	0b = Disabled 1b = Enabled
1	Self-Contained	0b	0b = Disabled 1b = Enabled
0	Automatic	0b	0b = Disabled 1b = Enabled

6.2.1.86 ComponentLocationOffset LSB (0x0073)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentLocationOffset	0x0	

6.2.1.87 ComponentLocationOffset MSB (0x0074)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentLocationOffset	0x0	

6.2.1.88 ComponentSize LSB (0x0075)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentSize LSB	0x0	

6.2.1.89 ComponentSize MSB (0x0076)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentSize MSB	0x0	

6.2.1.90 ComponentVersionStringTypeAndLength (0x0077)

Bit(s)	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

6.2.1.91 ComponentVersionString- Dev Starter Major (0x0078)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Dev Starter Major	0x3030	

6.2.1.92 ComponentVersionString - Dev Starter Minor (0x0079)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Dev Starter Minor	0x3030	

6.2.1.93 ComponentVersionString - EETRACK-ID MSB (0x007A)

Bit(s)	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).

6.2.1.94 ComponentVersionString - EETRACK-ID LSB (0x007B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - EETRACK-ID LSB	0x3030	

6.2.1.95 ComponentVersionString - Dot and Srev Byte 7 (0x007C)

Bit(s)	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	

6.2.1.96 ComponentVersionString - Srev Bytes 6-5 (0x007D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Srev Bytes 6-5	0x3030	

6.2.1.97 ComponentVersionString - Srev Bytes 4-3 (0x007E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Srev Bytes 4-3	0x3030	

6.2.1.98 ComponentVersionString - Srev Bytes 2-1 (0x007F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Srev Bytes 2-1	0x3030	

6.2.1.99 ComponentVersionString - Srev Byte 0 and Null (0x0080)

Bit(s)	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).

6.2.1.100 ComponentClassification (0x0081)

Bit(s)	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

6.2.1.101 ComponentIdentifier (0x0082)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentIdentifier	0x5	Valid values are: 0x1 = TLV Extension 0x2 = Link Topology Scratch Pad Area 0x3 = Descriptor Block 0x5 = OROM 0x6 = NVM 0x8 = Topology Netlist

6.2.1.102 ComponentComparisonStamp LSB (0x0083)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentComparisonStamp Low	0x0	

6.2.1.103 ComponentComparisonStamp MSB (0x0084)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentComparisonStamp High	0x0	

6.2.1.104 ComponentOptions (0x0085)

Bit(s)	Field Name	Default NVM Value	Description
15:2	Reserved	0x0	Reserved.
1	Use Component Comparison Stamp	1b	0b = Disabled 1b = Enabled
0	Force Update	1b	0b = Disabled 1b = Enabled

6.2.1.105 RequestedComponentActivationMethod (0x0086)

Bit(s)	Field Name	Default NVM Value	Description
15:6	Reserved	0x0	Reserved.
5	AC Power Cycle	0b	0b = Disabled 1b = Enabled
4	DC Power Cycle	0b	0b = Disabled 1b = Enabled
3	System Reboot	0b	0b = Disabled 1b = Enabled
2	Medium-Specific Reset	0b	0b = Disabled 1b = Enabled
1	Self-Contained	0b	0b = Disabled 1b = Enabled
0	Automatic	0b	0b = Disabled 1b = Enabled

6.2.1.106 ComponentLocationOffset LSB (0x0087)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentLocationOffset	0x0	

6.2.1.107 ComponentLocationOffset MSB (0x0088)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentLocationOffset	0x0	

6.2.1.108 ComponentSize LSB (0x0089)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentSize LSB	0x0	

6.2.1.109 ComponentSize MSB (0x008A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentSize MSB	0x0	

6.2.1.110 ComponentVersionStringTypeAndLength (0x008B)

Bit(s)	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

6.2.1.111 ComponentVersionString- CIVD High MSB (0x008C)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - CIVD High MSB	0x3030	

6.2.1.112 ComponentVersionString - CIVD High LSB (0x008D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - CIVD High LSB	0x3030	

6.2.1.113 ComponentVersionString - CIVD Low MSB (0x008E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - CIVD Low MSB	0x3030	

6.2.1.114 ComponentVersionString - CIVD Low LSB (0x008F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - CIVD Low LSB	0x3030	

6.2.1.115 ComponentVersionString - Dot and Srev Byte 7 (0x0090)

Bit(s)	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	

6.2.1.116 ComponentVersionString - Srev Bytes 6-5 (0x0091)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Srev Bytes 6-5	0x3030	

6.2.1.117 ComponentVersionString - Srev Bytes 4-3 (0x0092)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Srev Bytes 4-3	0x3030	

6.2.1.118 ComponentVersionString - Srev Bytes 2-1 (0x0093)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Srev Bytes 2-1	0x3030	

6.2.1.119 ComponentVersionString - Srev Byte 0 and Null (0x0094)

Bit(s)	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).

6.2.1.120 ComponentClassification (0x0095)

Bit(s)	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

6.2.1.121 ComponentIdentifier (0x0096)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentIdentifier	0x8	Valid values are: 0x1 = TLV Extension 0x2 = Link Topology Scratch Pad Area 0x3 = Descriptor Block 0x5 = OROM 0x6 = NVM 0x8 = Topology Netlist

6.2.1.122 ComponentComparisonStamp LSB (0x0097)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentComparisonStamp Low	0x0	

6.2.1.123 ComponentComparisonStamp MSB (0x0098)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentComparisonStamp High	0x0	

6.2.1.124 ComponentOptions (0x0099)

Bit(s)	Field Name	Default NVM Value	Description
15:2	Reserved	0x0	Reserved.
1	Use Component Comparison Stamp	0b	0b = Disabled 1b = Enabled
0	Force Update	0b	0b = Disabled 1b = Enabled

6.2.1.125 RequestedComponentActivationMethod (0x009A)

Bit(s)	Field Name	Default NVM Value	Description
15:6	Reserved		Reserved.
5	AC Power Cycle	0b	0b = Disabled 1b = Enabled
4	DC Power Cycle	0b	0b = Disabled 1b = Enabled
3	System Reboot	1b	0b = Disabled 1b = Enabled
2	Medium-Specific Reset	0b	0b = Disabled 1b = Enabled
1	Self-Contained	0b	0b = Disabled 1b = Enabled
0	Automatic	0b	0b = Disabled 1b = Enabled

6.2.1.126 ComponentLocationOffset LSB (0x009B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentLocationOffset	0x0	

6.2.1.127 ComponentLocationOffset MSB (0x009C)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentLocationOffset	0x0	

6.2.1.128 ComponentSize LSB (0x009D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentSize LSB	0x0	

6.2.1.129 ComponentSize MSB (0x009E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentSize MSB	0x0	

6.2.1.130 ComponentVersionStringTypeAndLength (0x009F)

Bit(s)	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

6.2.1.131 ComponentVersionString- ReleaseVersion Major Bytes 7-6 (0x00A0)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - ReleaseVersion Major Bytes 7-6	0x3030	

6.2.1.132 ComponentVersionString - ReleaseVersion Major Bytes 5-4 (0x00A1)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - ReleaseVersion Major Bytes 5-4	0x3030	

6.2.1.133 ComponentVersionString - ReleaseVersion Major Bytes 3-2 (0x00A2)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - ReleaseVersion Major Bytes 3-2	0x3030	

6.2.1.134 ComponentVersionString - ReleaseVersion Major Bytes 1-0 (0x00A3)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - ReleaseVersion Major Bytes 1-0	0x3030	

6.2.1.135 ComponentVersionString - Dot and ReleaseVersion Minor Byte 7 (0x00A4)

Bit(s)	Field Name	Default NVM Value	Description
15:8	ComponentVersionString -Dot and ReleaseVersion Minor Byte 7	0x30	
7:0	ComponentVersionString - Dot	0x2E	

6.2.1.136 ComponentVersionString - ReleaseVersion Minor Bytes 6-5 (0x00A5)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - ReleaseVersion Minor Bytes 6-5	0x3030	

6.2.1.137 ComponentVersionString - ReleaseVersion Minor Bytes 4-3 (0x00A6)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - ReleaseVersion Minor Bytes 4-3	0x3030	

6.2.1.138 ComponentVersionString - ReleaseVersion Minor Bytes 2-1 (0x00A7)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - ReleaseVersion Minor Bytes 2-1	0x3030	

6.2.1.139 ComponentVersionString - ReleaseVersion Minor Byte 0 and Dot (0x00A8)

Bit(s)	Field Name	Default NVM Value	Description
15:8	ComponentVersionString - Dot	0x2E	
7:0	ComponentVersionString -Dot and ReleaseVersion Minor Byte 0	0x30	

6.2.1.140 ComponentVersionString - ReleaseVersion Type Bytes 7-6 (0x00A9)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - ReleaseVersion Type Bytes 7-6	0x3030	

6.2.1.141 ComponentVersionString - ReleaseVersion Type Bytes 5-4 (0x00AA)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - ReleaseVersion Type Bytes 5-4	0x3030	

6.2.1.142 ComponentVersionString - ReleaseVersion Type Bytes 3-2 (0x00AB)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - ReleaseVersion Type Bytes 3-2	0x3030	

6.2.1.143 ComponentVersionString - ReleaseVersion Type Bytes 0-1 (0x00AC)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - ReleaseVersion Type Bytes 0-1	0x3030	

6.2.1.144 ComponentVersionString - Dot and Customer Netlist IANA Byte 7 (0x00AD)

Bit(s)	Field Name	Default NVM Value	Description
15:8	ComponentVersionString - Customer Netlist IANA Byte 7	0x30	
7:0	ComponentVersionString - Dot	0x2E	

6.2.1.145 ComponentVersionString - Customer Netlist IANA Bytes 6-5 (0x00AE)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Customer Netlist IANA Byte 6-5	0x3030	

6.2.1.146 ComponentVersionString - Customer Netlist IANA Bytes 4-3 (0x00AF)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Customer Netlist IANA Byte 4-3	0x3030	

6.2.1.147 ComponentVersionString - Customer Netlist IANA Bytes 2-1 (0x00B0)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Customer Netlist IANA Byte 2-1	0x3030	

6.2.1.148 ComponentVersionString - Customer Netlist IANA Byte 0 and Dot (0x00B1)

Bit(s)	Field Name	Default NVM Value	Description
15:8	ComponentVersionString - Dot	0x2E	
7:0	ComponentVersionString - Customer Netlist IANA Byte 0	0x30	

6.2.1.149 ComponentVersionString - Customer Netlist Version Bytes 3-2 (0x00B2)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Customer Netlist Version Bytes 3-2	0x3030	

6.2.1.150 ComponentVersionString - Customer Netlist Version Bytes 1-0 (0x00B3)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Customer Netlist Version Bytes 1-0	0x3030	

6.2.1.151 ComponentVersionString - Nulls (0x00B4)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Nulls	0x0	

6.2.1.152 PackageHeaderChecksum LSB (0x00B5)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PackageHeaderChecksum		CRC-32: Start Section -> Word: PLDM Header -> PackageHeaderIdentifier_0 End Section -> Word: PLDM Header -> ComponentVersionString

6.2.1.153 PackageHeaderChecksum MSB (0x00B6)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PackageHeaderChecksum		

6.3 NVM Content

Caution: In the tables of this section, contents of the NVM Default Value column might not reflect the value programmed in the specific NVM image provided to the customer.

6.3.1 NVM General Summary

Table 6-7. NVM General Summary Table

NVM Section	Section Reference
SPI Descriptor Section	6.3.2
Init Module Section	6.3.3
PFA Header Section	6.3.4
PFA Features Module Section	6.3.5
Feature Configuration Padding Module Section	6.3.6
PFA Immediate Values Module Section	6.3.7
Immediate Fields Padding Module Section	6.3.8
PFA VPD Module Section	6.3.9
PFA MNG Filter Section	6.3.10
PFA PT Configuration 0 Section	6.3.11
PFA PT Configuration 1 Section	6.3.12
PFA PT Configuration 2 Section	6.3.13
PFA PT Configuration 3 Section	6.3.14
PFA PT Configuration 4 Section	6.3.15
PFA PT Configuration 5 Section	6.3.16
PFA PT Configuration 6 Section	6.3.17
PFA PT Configuration 7 Section	6.3.18
Original EETrack ID Section	6.3.19
IBA Capabilities Module Section	6.3.20
PXE Setup Options Module Section	6.3.21
PXE Configuration Customization Options Module Section	6.3.22
PXE Version Module Section	6.3.23
VLAN Module Section	6.3.24
Boot Configuration Block Section	6.3.25
PBA Header Section	6.3.26
PBA Block Section	6.3.27
PCIR Registers PFA Auto-Load Module Section	6.3.28
POR Registers PFA Auto-Load Module Section	6.3.29
PSM Preserved Section	6.3.30
MinSrev Section	6.3.31

Table 6-7. NVM General Summary Table [continued]

NVM Section	Section Reference
PF MAC Address Section	6.3.32
MNG MAC Address Section	6.3.33
FW Logging Defaults Section	6.3.34
1588 Parameters Section	6.3.35
MD Link Topology Section	6.3.36
LLDP Preserved Section	6.3.37
RDE Module Section	6.3.38
Identical Content as PLDM Header ComponentImageSetVersionString Section	6.3.39
Software Checksum Module Section	6.3.40
RDMA Control Section	6.3.41
Link Default Override Mask Section	6.3.42
RDE Ethernet MTU Section	6.3.43
Default DCB Parameters Section	6.3.44
Current DCB Parameters Section	6.3.45
Padding Module Section	6.3.46
PCIR Type 1/2 Section	6.3.47
POR Type 1/2 Section	6.3.48
CORER Registers Auto-Load Module Section	6.3.49
GLOBR Registers Auto-Load Module Section	6.3.50
PE CORER Registers Section	6.3.51
Sideband Bus Auto-Load Section	6.3.52
EMP SR Settings Module Header Section	6.3.53
SR PF Allocations Section	6.3.54
LLDP Configuration Section	6.3.55
GFID Module Section	6.3.56
Manageability Module Header Section	6.3.57
Sideband Configuration Structure Section	6.3.58
OEM Section Section	6.3.59
Auto-Generated Pointers Module Section	6.3.60
NVM Image CSS Header Section	6.3.61
NVM Key and Signature Section	6.3.62
NVM Image Auth Header Section	6.3.63
SR1 - Should Be Copy of Shadow RAM: Section Clone	6.3.64
ML CSS Header Section	6.3.65
ML Key and Signature Section	6.3.66
ML Key and Signature - Signed Section	6.3.67
ML Auth Header Section	6.3.68

Table 6-7. NVM General Summary Table [continued]

NVM Section	Section Reference
Extended ML Header Section	6.3.69
ML Image Section	6.3.70
Analog PHY pre PLL Configuration Section	6.3.71
CSR Protected List Section	6.3.72
PCIe Analog Module Section	6.3.73
PCIR Registers Auto-Load Module Section	6.3.74
POR Registers Auto-Load Module Section	6.3.75
PCIR_PFA Auto-Load Whitelist Module Section	6.3.76
POR_PFA Auto-Load Whitelist Module Section	6.3.77
LVK Hashes Section	6.3.78
Recovery FW CSS Header Section	6.3.79
Recovery FW Key and Signature Section	6.3.80
Recovery FW Auth Header Section	6.3.81
Recovery FW Image Section	6.3.82
DCB Rx Module Section	6.3.83
DCB Tx Module Section	6.3.84
Ext. CORER Registers Auto-Load Module Section	6.3.85
EMP Global Module Section	6.3.86
EMP Settings Module Header Section	6.3.87
DL Scripts Section	6.3.88
Whitelist Section	6.3.89
Analog PHY Configuration Section	6.3.90
Configuration Metadata Section	6.3.91
Control Pipe Package Section	6.3.92
EMP Image Section	6.3.93
RDE Dictionaries Section	6.3.94
NVM Provisioning Area Section	6.3.95
OROM Section	6.3.96
OROM Provisioning Area Section	6.3.97
Link Topology Netlist Section	6.3.98
TLV Extension Provisioning Area Section	6.3.99
Link Topology Scratch Pad Area Section	6.3.100
FW Scratch Pad Area Section	6.3.101
Factory Settings Header Section	6.3.102
Factory Settings Area Section	6.3.103
Guarded Zone Section	6.3.104

6.3.2 SPI Descriptor Section

Table 6-8. SPI Descriptor Section Summary Table

Word Offset	Description	Section Reference
0x0000	SPI Flash Descriptor	6.3.2.1

6.3.2.1 SPI Flash Descriptor (0x0000)

Raw data module length: 2048 words

6.3.3 Init Module Section

This is the NVM header module that contains pointers to all other first-level sections. It also includes words that are relative to the whole NVM map.

Table 6-9. Init Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	NVM Control Word 1	6.3.3.1
0x0001	Non-Persistent End Pointer	6.3.3.2
0x0002	Last PFA Word Pointer	6.3.3.3
0x0003	GFID Pointer	6.3.3.4
0x0004 - 0x0006	Reserved	6.3.3.5
0x0007	Auto Generated Pointers Pointer	6.3.3.6
0x0008	Reserved	6.3.3.7
0x0009	EMP Global Module Pointer	6.3.3.8
0x000A	Guarded Zone Pointer	6.3.3.9
0x000B	EMP Image Pointer	6.3.3.10
0x000C - 0x000D	Reserved	6.3.3.11
0x000E	Manageability Module Pointer	6.3.3.12
0x000F	EMP Settings Module Pointer	6.3.3.13
0x0010	SW Compatibility Word 1	6.3.3.14
0x0011	SW Compatibility Word 2	6.3.3.15
0x0012	SW Compatibility Word 3	6.3.3.16
0x0013	SW Compatibility Word 4	6.3.3.17
0x0014	SW Compatibility Word 5	6.3.3.18
0x0015 - 0x0017	Reserved	6.3.3.19
0x0018	Software Reserved Word 1 - Dev Starter Version	6.3.3.20
0x0019	Software Reserved Word 2	6.3.3.21
0x001A	Software Reserved Word 3	6.3.3.22

Table 6-9. Init Module Section Summary Table [continued]

Word Offset	Description	Section Reference
0x001B	Software Reserved Word 4 - OEM Product Version Address Block Pointer	6.3.3.23
0x001C	Software Reserved Word 5	6.3.3.24
0x001D	Software Reserved Word 6	6.3.3.25
0x001E	Software Reserved Word 7	6.3.3.26
0x001F	Software Reserved Word 8	6.3.3.27
0x0020	Software Reserved Word 9	6.3.3.28
0x0021	Software Reserved Word 10	6.3.3.29
0x0022	Software Reserved Word 11	6.3.3.30
0x0023	Software Reserved Word 12	6.3.3.31
0x0024	Software Reserved Word 13	6.3.3.32
0x0025	Software Reserved Word 14	6.3.3.33
0x0026	Software Reserved Word 15	6.3.3.34
0x0027	Software Reserved Word 16	6.3.3.35
0x0028	Software Reserved Word 17	6.3.3.36
0x0029	Software Reserved Word 18 - Map Version	6.3.3.37
0x002A	Software Reserved Word 19 - NVM Image Version	6.3.3.38
0x002B	Software Reserved Word 20 - NVM Structure Version	6.3.3.39
0x002C	Software Reserved Word 21 - FCoE Offload	6.3.3.40
0x002D	Software Reserved Word 22 - EETRACK ID 1	6.3.3.41
0x002E	Software Reserved Word 23 - EETRACK ID 2	6.3.3.42
0x002F - 0x0033	Reserved	6.3.3.43
0x0034	Software Reserved Word 24 - Original EETRACK ID 1	6.3.3.44
0x0035	Software Reserved Word 25 - Original EETRACK ID 2	6.3.3.45
0x0036 - 0x003A	Reserved	6.3.3.46
0x003B	GLOBR Registers Auto-Load Pointer	6.3.3.47
0x003C	CORER Registers Auto-Load Pointer	6.3.3.48
0x003D	PHY Configuration Scripts (DNL) Pointer	6.3.3.49
0x003E - 0x003F	Reserved	6.3.3.50
0x0040	Preserved Field Area Pointer	6.3.3.51
0x0041	HLP SR Module Pointer	6.3.3.52
0x0042	1st NVM Bank Pointer	6.3.3.53
0x0043	NVM Bank Area Size	6.3.3.54
0x0044	1st OROM Bank Pointer	6.3.3.55
0x0045	OROM Bank Area Size	6.3.3.56
0x0046	1st TLV Extension Bank Pointer	6.3.3.57
0x0047	TLV Extension Bank Area Size	6.3.3.58
0x0048	EMP SR Settings Pointer	6.3.3.59

Table 6-9. Init Module Section Summary Table [continued]

Word Offset	Description	Section Reference
0x0049	Reserved	6.3.3.60
0x004A	PE CORER Registers Auto-Load Pointer	6.3.3.61
0x004B	Link Topology Scratch Pad Area Pointer	6.3.3.62
0x004C	Link Topology Scratch Pad Area Size	6.3.3.63
0x004D	Configuration Metadata Pointer	6.3.3.64
0x004E - 0x004F	Reserved	6.3.3.65
0x0050	FW Scratch Pad Area Pointer	6.3.3.66
0x0051	FW Scratch Pad Area Size	6.3.3.67
0x0052 - 0x0053	Reserved	6.3.3.68
0x0054	Analog PHY Configuration Module Pointer	6.3.3.69
0x0055	Soft SKUs	6.3.3.70
0x0056	Extended CORER Registers Auto-Load Pointer	6.3.3.71
0x0057	Recovery Firmware Pointer	6.3.3.72
0x0058	Control Pipe Package Pointer	6.3.3.73
0x0059 - 0x005A	Reserved	6.3.3.74
0x005B	DCB Rx Module Pointer	6.3.3.75
0x005C	DCB Tx Module Pointer	6.3.3.76
0x005D	Whitelist Pointer	6.3.3.77
0x005E	Sideband Auto-Load Pointer	6.3.3.78
0x005F	RDE Dictionaries Pointer	6.3.3.79
0x0060 - 0x0061	Reserved	6.3.3.80
0x0062	Factory Settings Size	6.3.3.81
0x0063 + 1*n, n=0...156	Spare NVM Header Words	6.3.3.82

6.3.3.1 NVM Control Word 1 (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:8	Reserved	0x0	Reserved.
7:6	Bank Validity	01b	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: 00b = Value 0x0 01b = Shadow Ram Bank Valid 10b = Value 0x2 11b = Value 0x3
5	Valid NVM Bank Index	0b	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: 0b = First Bank 1b = Second Bank
4	Valid TLV Extension Bank Index	0b	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: 0b = First Bank 1b = Second Bank
3	Valid OROM Bank Index	0b	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: 0b = First Bank 1b = Second Bank
2:0	Reserved	000b	Reserved.

6.3.3.2 Non-Persistent End Pointer (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	0b	Pointer Type: 0b = Word units. 1b = 4 KB sector units. Only the 4 KB sector unit is supported for this pointer.
14:0	Non-Persistent End Pointer	0x7FFF	Pointer to the end of the non-persistent Shadow RAM area

6.3.3.3 Last PFA Word Pointer (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	0b	Pointer Type: 0b = Word units. 1b = 4 KB sector units. Only the 4 KB sector unit is supported for this pointer.
14:0	Last PFA Word Pointer	0x0	Pointer to the end of the PFA excluding the PCI ALT module.

6.3.3.4 GFID Pointer (0x0003)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	0b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	PCIR GFID Module Pointer	0x0	Points to GFID Module Section. For GFID Module inner structure, see Section 6.3.56.

6.3.3.5 Reserved (0x0004 - 0x0006)

6.3.3.6 Auto-Generated Pointers Pointer (0x0007)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	0b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	Auto-Generated Pointers Module Pointer	0x0	Points to Auto-Generated Pointers Module Section. For Auto-Generated Pointers Module inner structure, see Section 6.3.60.

6.3.3.7 Reserved (0x0008)

6.3.3.8 EMP Global Module Pointer (0x0009)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	1b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	EMP Global Module Pointer	0x0	Points to EMP Global Module Section. For EMP Global Module inner structure, see Section 6.3.86.

6.3.3.9 Guarded Zone Pointer (0x000A)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	1b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	Guarded Zone Pointer	0x0	Points to Guarded Zone Section. For Guarded Zone inner structure, see Section 6.3.104.

6.3.3.10 EMP Image Pointer (0x000B)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	1b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	EMP Image Pointer	0x7FFF	Points to EMP Image Section. For EMP Image inner structure, see Section 6.3.93.

6.3.3.11 Reserved (0x000C - 0x000D)

6.3.3.12 Manageability Module Pointer (0x000E)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	0b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	Manageabilty Configuration Module Pointer	0x0	Points to Manageability Module Header Section. For Manageability Module Header inner structure, see Section 6.3.57.

6.3.3.13 EMP Settings Module Pointer (0x000F)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	1b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	EMP Module Pointer	0x0	Points to EMP Settings Module Header Section. For EMP Settings Module Header inner structure, see Section 6.3.87.

6.3.3.14 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.

Bit(s)	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	Reserved.
11	LOM	0b	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	1b	Legacy – not currently used. 0b = Client 1b = Server
9	Reserved	0b	Reserved.

Bit(s)	Field Name	Default NVM Value	Description
8	OEM/Retail	0b	Legacy – not currently used. 0b = Retail 1b = OEM
7:0	Reserved	0x0	Reserved.

6.3.3.15 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.

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.3.16 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.

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.3.17 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.

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.3.18 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.

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.3.19 Reserved (0x0015 - 0x0017)

6.3.3.20 Software Reserved Word 1 - Dev Starter Version (0x0018)

Dev_Starter map version used to produce this image. This word must be filled manually. Used for upgrade/downgrade algorithm.

Bit(s)	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.

6.3.3.21 Software Reserved Word 2 (0x0019)

Bit(s)	Field Name	Default NVM Value	Description
15:8	Reserved	0xFF	Reserved.
7	WoL Control Port 7	1b	Wake-on-LAN feature for Port 7. 0b = Supported and enabled. 1b = Disabled or not supported.
6	WoL Control Port 6	1b	Wake-on-LAN feature for Port 6. 0b = Supported and enabled. 1b = Disabled or not supported.
5	WoL Control Port 5	1b	Wake-on-LAN feature for Port 5. 0b = Supported and enabled. 1b = Disabled or not supported.
4	WoL Control Port 4	1b	Wake-on-LAN feature for Port 4. 0b = Supported and enabled. 1b = Disabled or not supported.
3	WoL Control Port 3	1b	Wake-on-LAN feature for Port 3. 0b = Supported and enabled. 1b = Disabled or not supported.
2	WoL Control Port 2	1b	Wake-on-LAN feature for Port 2. 0b = Supported and enabled. 1b = Disabled or not supported.
1	WoL Control Port 1	1b	Wake-on-LAN feature for Port 1. 0b = Supported and enabled. 1b = Disabled or not supported.
0	WoL Control Port 0	1b	Wake-on-LAN feature for Port 0. 0b = Supported and enabled. 1b = Disabled or not supported.

6.3.3.22 Software Reserved Word 3 (0x001A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.3.23 Software Reserved Word 4 - OEM Product Version Address Block Pointer (0x001B)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	0b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	OEM Product Version Address Block Pointer	0x0	

6.3.3.24 Software Reserved Word 5 (0x001C)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.3.25 Software Reserved Word 6 (0x001D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.3.26 Software Reserved Word 7 (0x001E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.3.27 Software Reserved Word 8 (0x001F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.3.28 Software Reserved Word 9 (0x0020)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.3.29 Software Reserved Word 10 (0x0021)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.3.30 Software Reserved Word 11 (0x0022)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.3.31 Software Reserved Word 12 (0x0023)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.3.32 Software Reserved Word 13 (0x0024)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.3.33 Software Reserved Word 14 (0x0025)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.3.34 Software Reserved Word 15 (0x0026)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.3.35 Software Reserved Word 16 (0x0027)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.3.36 Software Reserved Word 17 (0x0028)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.3.37 Software Reserved Word 18 - Map Version (0x0029)

Automatically generated by the tool.

Bit(s)	Field Name	Default NVM Value	Description
15:0	Map Version		

6.3.3.38 Software Reserved Word 19 - NVM Image Version (0x002A)

Automatically generated by the tool.

Bit(s)	Field Name	Default NVM Value	Description
15:0	NVM Image Version		

6.3.3.39 Software Reserved Word 20 - NVM Structure Version (0x002B)

Bit(s)	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.

6.3.3.40 Software Reserved Word 21 - FCoE Offload (0x002C)

This word is for Platform/NIC/LOM specific settings.

Bit(s)	Field Name	Default NVM Value	Description
15:2	Reserved	0x3FFF	Reserved.
1	FCoE Offload	1b	This bit indicates to software if the device supports FCoE Offload. 0b = FCoE Offload Enabled. 1b = FCoE Offload Disabled.
0	Reserved	1b	Reserved.

6.3.3.41 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.

Bit(s)	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.

6.3.3.42 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.

Bit(s)	Field Name	Default NVM Value	Description
15:0	eTrack_ID Word 2		

6.3.3.43 Reserved (0x002F - 0x0033)

6.3.3.44 Software Reserved Word 24 - Original EETRACK ID 1 (0x0034)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Original EETRACK ID 1	0x0	

6.3.3.45 Software Reserved Word 25 - Original EETRACK ID 2 (0x0035)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Original EETRACK ID 2	0x0	

6.3.3.46 Reserved (0x0036 - 0x003A)

6.3.3.47 GLOBR Registers Auto-Load Pointer (0x003B)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	0b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	GLOBR Registers Auto-Load Module Pointer	0x7FFF	Points to GLOBR Registers Auto-Load Module Section. For GLOBR Registers Auto-Load Module inner structure, see Section 6.3.50.

6.3.3.48 CORER Registers Auto-Load Pointer (0x003C)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	0b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	CORER Registers Auto-Load Module Pointer	0x7FFF	Points to CORER Registers Auto-Load Module Section. For CORER Registers Auto-Load Module inner structure, see Section 6.3.49.

6.3.3.49 PHY Configuration Scripts (DNL) Pointer (0x003D)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	1b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	DNL Pointer	0x7FFF	Points to DL Scripts Section. For DL Scripts inner structure, see Section 6.3.88.

6.3.3.50 Reserved (0x003E - 0x003F)

6.3.3.51 Preserved Field Area Pointer (0x0040)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	0b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	PFA Pointer	0x7FFF	Points to PFA Header Section. For PFA Header inner structure, see Section 6.3.4.

6.3.3.52 HLP SR Module Pointer (0x0041)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	0b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	HLP Module Pointer	0x0	

6.3.3.53 1st NVM Bank Pointer (0x0042)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	1b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	1st NVM Bank Pointer	0x7FFF	

6.3.3.54 NVM Bank Area Size (0x0043)

Bit(s)	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	Reserved.
11:0	NVM Bank Area Size	0x41A	Size expressed in 4 KB sector units.

6.3.3.55 1st OROM Bank Pointer (0x0044)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	1b	Pointer Type: 0b = Word units. 1b = 4 KB 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. Points to OROM Section. For OROM inner structure, see Section 6.3.96.

6.3.3.56 OROM Bank Area Size (0x0045)

Bit(s)	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	Reserved.
11:0	OROM Bank Area Size	0x7D	Size expressed in 4 KB sector units.

6.3.3.57 1st TLV Extension Bank Pointer (0x0046)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	1b	Pointer Type: 0b = Word units. 1b = 4 KB 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. Points to Link Topology Netlist Section. For Link Topology Netlist inner structure, see Section 6.3.98.

6.3.3.58 TLV Extension Bank Area Size (0x0047)

Bit(s)	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	Reserved.
11:0	TLV Extension Bank Area Size	0x7	Size expressed in 4 KB sector units.

6.3.3.59 EMP SR Settings Pointer (0x0048)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	0b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	EMP SR Settings Pointer	07FFF	Points to EMP SR Settings Module Header Section. For EMP SR Settings Module Header inner structure, see Section 6.3.53.

6.3.3.60 Reserved (0x0049)

6.3.3.61 PE CORER Registers Auto-Load Pointer (0x004A)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	0b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	PE CORER Config Pointer	07FFF	Points to PE CORER Registers Section. For PE CORER Registers inner structure, see Section 6.3.51.

6.3.3.62 Link Topology Scratch Pad Area Pointer (0x004B)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	1b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	Link Topology Scratch Pad Area Pointer	07FFF	Points to Link Topology Scratch Pad Area Section. For Link Topology Scratch Pad Area inner structure, see Section 6.3.100.

6.3.3.63 Link Topology Scratch Pad Area Size (0x004C)

Bit(s)	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	Reserved.
11:0	Link Topology Scratch Pad Area Size	0x5	Size expressed in 4 KB sector units.

6.3.3.64 Configuration Metadata Pointer (0x004D)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	1b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	Configuration Metadata Pointer	07FFF	Points to Configuration Metadata Section. For Configuration Metadata inner structure, see Section 6.3.91.

6.3.3.65 Reserved (0x004E - 0x004F)

6.3.3.66 FW Scratch Pad Area Pointer (0x0050)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	1b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	FW Scratch Pad Area Pointer	07FFF	Points to FW Scratch Pad Area Section. For FW Scratch Pad Area inner structure, see Section 6.3.101.

6.3.3.67 FW Scratch Pad Area Size (0x0051)

Bit(s)	Field Name	Default NVM Value	Description
15:10	Reserved	0x0	Reserved.
9:0	FW Scratch Pad Area Size	0x40	Size expressed in 4 KB sector units.

6.3.3.68 Reserved (0x0052 - 0x0053)

6.3.3.69 Analog PHY Configuration Module Pointer (0x0054)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	1b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	Internal PHY Configuration Module Pointer	07FFF	Points to Analog PHY Configuration Section. For Analog PHY Configuration inner structure, see Section 6.3.90.

6.3.3.70 Soft SKUs (0x0055)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	Reserved.

6.3.3.71 Extended CORER Registers Auto-Load Pointer (0x0056)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	1b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	Extended CORER Registers Auto-Load Module Pointer	07FFF	Points to Ext. CORER Registers Auto-Load Module Section. For Ext. CORER Registers Auto-Load Module inner structure, see Section 6.3.85.

6.3.3.72 Recovery Firmware Pointer (0x0057)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	1b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	Recovery Firmware Pointer	0x0	Pointer to Recovery Firmware section.

6.3.3.73 Control Pipe Package Pointer (0x0058)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	1b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	Control Pipe Package Pointer	07FFF	Points to Control Pipe Package Section. For Control Pipe Package inner structure, see Section 6.3.92.

6.3.3.74 Reserved (0x0059 - 0x005A)

6.3.3.75 DCB Rx Module Pointer (0x005B)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	1b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	DCB Rx Module Pointer	07FFF	Points to DCB Rx Module Section. For DCB Rx Module inner structure, see Section 6.3.83.

6.3.3.76 DCB Tx Module Pointer (0x005C)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	1b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	DCB Tx Module Pointer	07FFF	Points to DCB Tx Module Section. For DCB Tx Module inner structure, see Section 6.3.84.

6.3.3.77 Whitelist Pointer (0x005D)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	1b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	White List Pointer	07FFF	Points to White list Section. For White list inner structure, see Section 6.3.89.

6.3.3.78 Sideband Auto-Load Pointer (0x005E)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	0b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	Sideband Auto-Load Pointer	07FFF	Points to Sideband Bus Auto-Load Section. For Sideband Bus Auto-Load inner structure, see Section 6.3.52.

6.3.3.79 RDE Dictionaries Pointer (0x005F)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type	1b	Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	RDE Dictionaries Pointer	07FFF	Points to RDE Dictionaries Section. For RDE Dictionaries inner structure, see Section 6.3.94.

6.3.3.80 Reserved (0x0060 - 0061)

6.3.3.81 Factory Settings Size (0x0062)

Bit(s)	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	Reserved.
11:0	Factory Settings Size	0xD	Size expressed in 4 KB sector units.

6.3.3.82 Spare NVM Header Words[n] (0x0063 + 1*n, n=0...156)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.4 PFA Header Section

Preserved fields area header.

Table 6-10. PFA Header Section Summary Table

Word Offset	Description	Section Reference
0x0000	PFA Length	6.3.4.1

6.3.4.1 PFA Length (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PFA Length	0x2F00	

6.3.5 PFA Features Module Section

Table 6-11. PFA Features Module Section Summary Table

Word Offset	Description	Section Reference
0x0101	Sub Module Type - Features	6.3.5.1

6.3.5.1 Sub Module Type - Features (0x0101)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x49	Valid values are: 0x49 = Feature Configuration

6.3.6 Feature Configuration Padding Module Section

Table 6-12. Feature Configuration Padding Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type - Padding	6.3.6.1
0x0001	Length	6.3.6.2
0x0002	Padding	6.3.6.3

6.3.6.1 Sub Module Type - Padding (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0xFFFF	Valid values are: 0xFFFF = Padding Module

6.3.6.2 Length (0x0001)

Bit(s)	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

6.3.6.3 Padding (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:0	[New Field]		

6.3.7 PFA Immediate Values Module Section

Table 6-13. PFA Immediate Values Module Section Summary Table

Word Offset	Description	Section Reference
0x0901	Sub Module Type - Immediate	6.3.7.1

6.3.7.1 Sub Module Type - Immediate (0x0901)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x4E	Valid values are: 0x4E = Immediate Configuration

6.3.8 Immediate Fields Padding Module Section

Table 6-14. Immediate Fields Padding Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type Padding	6.3.8.1
0x0001	Length	6.3.8.2
0x0002	Padding	6.3.8.3

6.3.8.1 Sub Module Type Padding (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0xFFFF	Valid values are: 0xFFFF = Padding Module

6.3.8.2 Length (0x0001)

Bit(s)	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

6.3.8.3 Padding (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:0	[New Field]	0xFFFF	

6.3.9 PFA VPD Module Section

VPD area. Vital product data loaded by OEM. It contains RO and RW info about the NIC/LOM.

Table 6-15. PFA VPD Module Section Summary Table

Word Offset	Description	Section Reference
0x0F01	Sub Module Type - VPD	6.3.9.1
0x0F02	Length	6.3.9.2
0x0F03	VPD Data	6.3.9.3

6.3.9.1 Sub Module Type - VPD (0x0F01)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x2F	Valid values are: 0x2F = VPD Module

6.3.9.2 Length (0x0F02)

Bit(s)	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

6.3.9.3 VPD Data (0x0F03)

Raw data module length: 512 words

6.3.10 PFA MNG Filter Section

Table 6-16. PFA MNG Filter Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type - MNG Filter	6.3.10.1
0x0001	Section Length	6.3.10.2
0x0002	Flexible Filter Data	6.3.10.3

6.3.10.1 Sub Module Type - MNG Filter (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x107	Valid values are: 0x107 = MNG Filters

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.

Bit(s)	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

6.3.10.3 Flexible Filter Data (0x0002)

Raw data module length: variable

6.3.11 PFA PT Configuration 0 Section

Pass-through filters of Port 0 PFA section header.

Table 6-17. PFA PT Configuration 0 Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type PT Module 0	6.3.11.1
0x0001	Section Length	6.3.11.2
0x0002 + 2*n, n=0...3	LAN IPv4 Address (LSB) MIPAF0	6.3.11.3
0x0003 + 2*n, n=0...3	LAN IPv4 Address (MSB) MIPAF0	6.3.11.4
0x000A + 2*n, n=0...15	LAN Flexible Filter Port	6.3.11.5
0x000B + 2*n, n=0...15	LAN Flexible Filter Port - Modifiers	6.3.11.6
0x002A + 1*n, n=0...7	LAN VLAN Filter	6.3.11.7
0x0032	LAN MANC Value LSB	6.3.11.8
0x0033	LAN MANC Value MSB	6.3.11.9
0x0034	Receive Enable 1 - LRXEN1	6.3.11.10
0x0035	Receive Enable 2 - LRXEN2	6.3.11.11
0x0036	LAN MNGONLY LSB	6.3.11.12
0x0037	LAN MNGONLY MSB	6.3.11.13
0x0038 + 4*n, n=0...6	Manageability Decision Filters LSB	6.3.11.14
0x0039 + 4*n, n=0...6	Manageability Decision Filters MSB	6.3.11.15
0x003A + 4*n, n=0...6	Manageability Extended Decision Filters LSB	6.3.11.16
0x003B + 4*n, n=0...6	Manageability Extended Decision Filters MSB	6.3.11.17
0x0054 + 2*n, n=0...3	Manageability EtherType Filter (METF) LSB	6.3.11.18
0x0055 + 2*n, n=0...3	Manageability EtherType Filter (METF) MSB	6.3.11.19
0x005C	ARP Response IPv4 Address LSB	6.3.11.20
0x005D	ARP Response IPv4 Address MSB	6.3.11.21
0x005E + 8*n, n=0...3	IPv6 Address Bytes 0-1	6.3.11.22
0x005F + 8*n, n=0...3	IPv6 Address Bytes 2-3	6.3.11.23
0x0060 + 8*n, n=0...3	IPv6 Address Bytes 4-5	6.3.11.24
0x0061 + 8*n, n=0...3	IPv6 Address Bytes 6-7	6.3.11.25
0x0062 + 8*n, n=0...3	IPv6 Address Bytes 8-9	6.3.11.26
0x0063 + 8*n, n=0...3	IPv6 Address Bytes 10-11	6.3.11.27
0x0064 + 8*n, n=0...3	IPv6 Address Bytes 12-13	6.3.11.28
0x0065 + 8*n, n=0...3	IPv6 Address Bytes 14-15	6.3.11.29
0x007E	Manageability Special Modifiers LSB	6.3.11.30
0x007F	Manageability Special Modifiers MSB	6.3.11.31

6.3.11.1 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

6.3.11.2 Section Length (0x0001)

The length of the section in words. Note that section length does not include a count for the section length word.

Bit(s)	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 -> Manageability Special Modifiers MSB

6.3.11.3 LAN IPv4 Address (LSB) MIPAF0[n] (0x0002 + 2*n, n=0...3)

Bit(s)	Field Name	Default NVM Value	Description
15:8	IPv4 Address, Byte 1	0xFF	
7:0	IPv4 Address, Byte 0	0xFF	

6.3.11.4 LAN IPv4 Address (MSB) MIPAF0[n] (0x0003 + 2*n, n=0...3)

Bit(s)	Field Name	Default NVM Value	Description
15:8	IPv4 Address, Byte 3	0xFF	
7:0	IPv4 Address, Byte 4	0xFF	

6.3.11.5 LAN Flexible Filter Port[n] (0x000A + 2*n, n=0...15)

Bit(s)	Field Name	Default NVM Value	Description
15:0	LAN UDP Flexible Filter Port0	0xFFFF	

6.3.11.6 LAN Flexible Filter Port - Modifiers[n] (0x000B + 2*n, n=0...15)

Bit(s)	Field Name	Default NVM Value	Description
15:3	Reserved		Reserved.
2	Source Destination	0b	0b = Destination 1b = Source
1	Accept TCP	0b	If set, filter match if packet is TCP. 0b = Filter 1b = Accept
0	Accept UDP	0b	If set, filter match if packet is UDP. 0b = Filter 1b = Accept

6.3.11.7 LAN VLAN Filter[n] (0x002A + 1*n, n=0...7)

Bit(s)	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	Reserved.
11:0	VLAN Filter 0 Value	0xFFF	

6.3.11.8 LAN MANC Value LSB (0x0032)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	Reserved.

6.3.11.9 LAN MANC Value MSB (0x0033)

Bit(s)	Field Name	Default NVM Value	Description
15:11	Reserved	0x0	Reserved.
10	NET_TYPE	0b	
9	FIXED_NET_TYPE	0b	
8	Reserved	0b	Reserved.
7	EN_XSUM_FILTER	0b	Should be used to set the RBCR.*EN_XSUM_FILTERn* bit, where *n* is the port number).
6:0	Reserved	0x0	Reserved.

6.3.11.10 Receive Enable 1 - LRXEN1 (0x0034)

Bit(s)	Field Name	Default NVM Value	Description
15:9	Receive Enable Byte 12	0x0	BMC SMBus slave address.
8	Reserved	0b	Reserved.
7	Enable BMC Dedicated	0b	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	1b	Reserved.
5:4	Notification Method	00b	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	0b	When set, the firmware offloads 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	0b	
1	Enable Receive All	0b	
0	Enable Receive TCO	0b	

6.3.11.11 Receive Enable 2 - LRXEN2 (0x0035)

Bit(s)	Field Name	Default NVM Value	Description
15:8	Receive Enable Byte 14	0x0	Alert value.
7:0	Receive Enable Byte 13	0x0	Interface data.

6.3.11.12 LAN MNGONLY LSB (0x0036)

Bit(s)	Field Name	Default NVM Value	Description
15:8	Reserved	0x0	Reserved.
7:0	Exclusive to MNG	0x0	

6.3.11.13 LAN MNGONLY MSB (0x0037)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	Reserved.

6.3.11.14 Manageability Decision Filters LSB[n] (0x0038 + 4*n, n=0...6)

Bit(s)	Field Name	Default NVM Value	Description
15:0	MDEF0_L	0x0	

6.3.11.15 Manageability Decision Filters MSB[n] (0x0039 + 4*n, n=0...6)

Bit(s)	Field Name	Default NVM Value	Description
15:0	MDEF0_M	0x0	

6.3.11.16 Manageability Extended Decision Filters LSB[n] (0x003A + 4*n, n=0...6)

Bit(s)	Field Name	Default NVM Value	Description
15:0	MDEFEXT0_L	0x0	

6.3.11.17 Manageability Extended Decision Filters MSB[n] (0x003B + 4*n, n=0...6)

Bit(s)	Field Name	Default NVM Value	Description
15:0	MDEFEXT0_M	0x0	

6.3.11.18 Manageability EtherType Filter (METF) LSB[n] (0x0054 + 2*n, n=0...3)

Bit(s)	Field Name	Default NVM Value	Description
15:0	METF0_L	0x0	Loaded to 16 LS bits of METF[0] register.

6.3.11.19 Manageability EtherType Filter (METF) MSB[n] (0x0055 + 2*n, n=0...3)

Bit(s)	Field Name	Default NVM Value	Description
15:0	METF0_M	0x0	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).

6.3.11.20 ARP Response IPv4 Address LSB (0x005C)

Bit(s)	Field Name	Default NVM Value	Description
15:8	Byte 1	0x0	Firmware use.
7:0	Byte 0	0x0	Firmware use.

6.3.11.21 ARP Response IPv4 Address MSB (0x005D)

Bit(s)	Field Name	Default NVM Value	Description
15:8	Byte 1	0x0	Firmware use.
7:0	Byte 0	0x0	Firmware use.

6.3.11.22 IPv6 Address Bytes 0-1[n] (0x005E + 8*n, n=0...3)

Bit(s)	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 0-1	0x0	

6.3.11.23 IPv6 Address Bytes 2-3[n] (0x005F + 8*n, n=0...3)

Bit(s)	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 2-3	0x0	

6.3.11.24 IPv6 Address Bytes 4-5[n] (0x0060 + 8*n, n=0...3)

Bit(s)	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 4-5	0x0	

6.3.11.25 IPv6 Address Bytes 6-7[n] (0x0061 + 8*n, n=0...3)

Bit(s)	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 6-7	0x0	

6.3.11.26 IPv6 Address Bytes 8-9[n] (0x0062 + 8*n, n=0...3)

Bit(s)	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 8-9	0x0	

6.3.11.27 IPv6 Address Bytes 10-11[n] (0x0063 + 8*n, n=0...3)

Bit(s)	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 10-11	0x0	

6.3.11.28 IPv6 Address Bytes 12-13[n] (0x0064 + 8*n, n=0...3)

Bit(s)	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 12-13	0x0	

6.3.11.29 IPv6 Address Bytes 14-15[n] (0x0065 + 8*n, n=0...3)

Bit(s)	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 14-15	0x0	

6.3.11.30 Manageability Special Modifiers LSB (0x007E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	MSFM_L	0x000F	

6.3.11.31 Manageability Special Modifiers MSB (0x007F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	MSFM_M	0x0	

6.3.12 PFA PT Configuration 1 Section

Pass-through filters of Port 1 PFA section header.

Table 6-18. PFA PT Configuration 1 Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type PT Module 1	6.3.12.1
0x0001	Section Length	6.3.12.2
0x0002 + 2*n, n=0...3	LAN IPv4 Address (LSB) MIPAF0	6.3.12.3
0x0003 + 2*n, n=0...3	LAN IPv4 Address (MSB) MIPAF0	6.3.12.4
0x000A + 2*n, n=0...15	LAN Flexible Filter Port	6.3.12.5
0x000B + 2*n, n=0...15	LAN Flexible Filter Port - Modifiers	6.3.12.6
0x002A + 1*n, n=0...7	LAN VLAN Filter	6.3.12.7
0x0032	LAN MANC Value LSB	6.3.12.8
0x0033	LAN MANC Value MSB	6.3.12.9
0x0034	Receive Enable 1 - LRXEN1	6.3.12.10
0x0035	Receive Enable 2 - LRXEN2	6.3.12.11
0x0036	LAN MNGONLY LSB	6.3.12.12
0x0037	LAN MNGONLY MSB	6.3.12.13
0x0038 + 4*n, n=0...6	Manageability Decision Filters LSB	6.3.12.14
0x0039 + 4*n, n=0...6	Manageability Decision Filters MSB	6.3.12.15
0x003A + 4*n, n=0...6	Manageability Extended Decision Filters LSB	6.3.12.16
0x003B + 4*n, n=0...6	Manageability Extended Decision Filters MSB	6.3.12.17
0x0054 + 2*n, n=0...3	Manageability EtherType Filter (METF) LSB	6.3.12.18
0x0055 + 2*n, n=0...3	Manageability EtherType Filter (METF) MSB	6.3.12.19
0x005C	ARP Response IPv4 Address LSB	6.3.12.20
0x005D	ARP Response IPv4 Address MSB	6.3.12.21
0x005E + 8*n, n=0...3	IPv6 Address Bytes 0-1	6.3.12.22
0x005F + 8*n, n=0...3	IPv6 Address Bytes 2-3	6.3.12.23
0x0060 + 8*n, n=0...3	IPv6 Address Bytes 4-5	6.3.12.24
0x0061 + 8*n, n=0...3	IPv6 Address Bytes 6-7	6.3.12.25
0x0062 + 8*n, n=0...3	IPv6 Address Bytes 8-9	6.3.12.26
0x0063 + 8*n, n=0...3	IPv6 Address Bytes 10-11	6.3.12.27
0x0064 + 8*n, n=0...3	IPv6 Address Bytes 12-13	6.3.12.28
0x0065 + 8*n, n=0...3	IPv6 Address Bytes 14-15	6.3.12.29
0x007E	Manageability Special Modifiers LSB	6.3.12.30
0x007F	Manageability Special Modifiers MSB	6.3.12.31

6.3.12.1 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

6.3.12.2 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, see Section 6.3.11.2.

6.3.12.3 LAN IPv4 Address (LSB) MIPAF0[n] (0x0002 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.3.

6.3.12.4 LAN IPv4 Address (MSB) MIPAF0[n] (0x0003 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.4.

6.3.12.5 LAN Flexible Filter Port[n] (0x000A + 2*n, n=0...15)

For inner structure, see Section 6.3.11.5.

6.3.12.6 LAN Flexible Filter Port - Modifiers[n] (0x000B + 2*n, n=0...15)

For inner structure, see Section 6.3.11.6.

6.3.12.7 LAN VLAN Filter[n] (0x002A + 1*n, n=0...7)

For inner structure, see Section 6.3.11.7.

6.3.12.8 LAN MANC Value LSB (0x0032)

For inner structure, see Section 6.3.11.8.

6.3.12.9 LAN MANC Value MSB (0x0033)

For inner structure, see Section 6.3.11.9.

6.3.12.10 Receive Enable 1 - LRXEN1 (0x0034)

For inner structure, see Section 6.3.11.10.

6.3.12.11 Receive Enable 2 - LRXEN2 (0x0035)

For inner structure, see Section 6.3.11.11.

6.3.12.12 LAN MNGONLY LSB (0x0036)

For inner structure, see Section 6.3.11.12.

6.3.12.13 LAN MNGONLY MSB (0x0037)

For inner structure, see Section 6.3.11.13.

6.3.12.14 Manageability Decision Filters LSB[n] (0x0038 + 4*n, n=0...6)

For inner structure, see Section 6.3.11.14.

6.3.12.15 Manageability Decision Filters MSB[n] (0x0039 + 4*n, n=0...6)

For inner structure, see Section 6.3.11.15.

6.3.12.16 Manageability Extended Decision Filters LSB[n] (0x003A + 4*n, n=0...6)

For inner structure, see Section 6.3.11.16.

6.3.12.17 Manageability Extended Decision Filters MSB[n] (0x003B + 4*n, n=0...6)

For inner structure, see Section 6.3.11.17.

6.3.12.18 Manageability EtherType Filter (METF) LSB[n] (0x0054 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.18.

6.3.12.19 Manageability EtherType Filter (METF) MSB[n] (0x0055 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.19.

6.3.12.20 ARP Response IPv4 Address LSB (0x005C)

For inner structure, see Section 6.3.11.20.

6.3.12.21 ARP Response IPv4 Address MSB (0x005D)

For inner structure, see Section 6.3.11.21.

6.3.12.22 IPv6 Address Bytes 0-1[n] (0x005E + 8*n, n=0...3)

For inner structure, see Section 6.3.11.22.

6.3.12.23 IPv6 Address Bytes 2-3[n] (0x005F + 8*n, n=0...3)

For inner structure, see Section 6.3.11.23.

6.3.12.24 IPv6 Address Bytes 4-5[n] (0x0060 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.24.

6.3.12.25 IPv6 Address Bytes 6-7[n] (0x0061 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.25.

6.3.12.26 IPv6 Address Bytes 8-9[n] (0x0062 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.26.

6.3.12.27 IPv6 Address Bytes 10-11[n] (0x0063 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.27.

6.3.12.28 IPv6 Address Bytes 12-13[n] (0x0064 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.28.

6.3.12.29 IPv6 Address Bytes 14-15[n] (0x0065 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.29.

6.3.12.30 Manageability Special Modifiers LSB (0x007E)

For inner structure, see Section 6.3.11.30.

6.3.12.31 Manageability Special Modifiers MSB (0x007F)

For inner structure, see Section 6.3.11.31.

6.3.13 PFA PT Configuration 2 Section

Pass-through filters of Port 2 PFA section header.

Table 6-19. PFA PT Configuration 2 Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type PT Module 2	6.3.13.1
0x0001	Section Length	6.3.13.2
0x0002 + 2*n, n=0...3	LAN IPv4 Address (LSB) MIPAF0	6.3.13.3
0x0003 + 2*n, n=0...3	LAN IPv4 Address (MSB) MIPAF0	6.3.13.4
0x000A + 2*n, n=0...15	LAN Flexible Filter Port	6.3.13.5
0x000B + 2*n, n=0...15	LAN Flexible Filter Port - Modifiers	6.3.13.6
0x002A + 1*n, n=0...7	LAN VLAN Filter	6.3.13.7
0x0032	LAN MANC Value LSB	6.3.13.8
0x0033	LAN MANC Value MSB	6.3.13.9
0x0034	Receive Enable 1 - LRXEN1	6.3.13.10
0x0035	Receive Enable 2 - LRXEN2	6.3.13.11
0x0036	LAN MNGONLY LSB	6.3.13.12
0x0037	LAN MNGONLY MSB	6.3.13.13
0x0038 + 4*n, n=0...6	Manageability Decision Filters LSB	6.3.13.14
0x0039 + 4*n, n=0...6	Manageability Decision Filters MSB	6.3.13.15
0x003A + 4*n, n=0...6	Manageability Extended Decision Filters LSB	6.3.13.16
0x003B + 4*n, n=0...6	Manageability Extended Decision Filters MSB	6.3.13.17
0x0054 + 2*n, n=0...3	Manageability EtherType Filter (METF) LSB	6.3.13.18
0x0055 + 2*n, n=0...3	Manageability EtherType Filter (METF) MSB	6.3.13.19
0x005C	ARP Response IPv4 Address LSB	6.3.13.20
0x005D	ARP Response IPv4 Address MSB	6.3.13.21
0x005E + 8*n, n=0...3	IPv6 Address Bytes 0-1	6.3.13.22
0x005F + 8*n, n=0...3	IPv6 Address Bytes 2-3	6.3.13.23
0x0060 + 8*n, n=0...3	IPv6 Address Bytes 4-5	6.3.13.24
0x0061 + 8*n, n=0...3	IPv6 Address Bytes 6-7	6.3.13.25
0x0062 + 8*n, n=0...3	IPv6 Address Bytes 8-9	6.3.13.26
0x0063 + 8*n, n=0...3	IPv6 Address Bytes 10-11	6.3.13.27
0x0064 + 8*n, n=0...3	IPv6 Address Bytes 12-13	6.3.13.28
0x0065 + 8*n, n=0...3	IPv6 Address Bytes 14-15	6.3.13.29
0x007E	Manageability Special Modifiers LSB	6.3.13.30
0x007F	Manageability Special Modifiers MSB	6.3.13.31

6.3.13.1 Sub Module Type PT Module 2 (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x10A	Valid values are: 0x10A = Pass-Through Control Words Structure 2

6.3.13.2 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, see Section 6.3.11.2.

6.3.13.3 LAN IPv4 Address (LSB) MIPAF0[n] (0x0002 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.3.

6.3.13.4 LAN IPv4 Address (MSB) MIPAF0[n] (0x0003 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.4.

6.3.13.5 LAN Flexible Filter Port[n] (0x000A + 2*n, n=0...15)

For inner structure, see Section 6.3.11.5.

6.3.13.6 LAN Flexible Filter Port - Modifiers[n] (0x000B + 2*n, n=0...15)

For inner structure, see Section 6.3.11.6.

6.3.13.7 LAN VLAN Filter[n] (0x002A + 1*n, n=0...7)

For inner structure, see Section 6.3.11.7.

6.3.13.8 LAN MANC Value LSB (0x0032)

For inner structure, see Section 6.3.11.8.

6.3.13.9 LAN MANC Value MSB (0x0033)

For inner structure, see Section 6.3.11.9.

6.3.13.10 Receive Enable 1 - LRXEN1 (0x0034)

For inner structure, see Section 6.3.11.10.

6.3.13.11 Receive Enable 2 - LRXEN2 (0x0035)

For inner structure, see Section 6.3.11.11.

6.3.13.12 LAN MNGONLY LSB (0x0036)

For inner structure, see Section 6.3.11.12.

6.3.13.13 LAN MNGONLY MSB (0x0037)

For inner structure, see Section 6.3.11.13.

6.3.13.14 Manageability Decision Filters LSB[n] (0x0038 + 4*n, n=0...6)

For inner structure, see Section 6.3.11.14.

6.3.13.15 Manageability Decision Filters MSB[n] (0x0039 + 4*n, n=0...6)

For inner structure, see Section 6.3.11.15.

6.3.13.16 Manageability Extended Decision Filters LSB[n] (0x003A + 4*n, n=0...6)

For inner structure, see Section 6.3.11.16.

6.3.13.17 Manageability Extended Decision Filters MSB[n] (0x003B + 4*n, n=0...6)

For inner structure, see Section 6.3.11.17.

6.3.13.18 Manageability EtherType Filter (METF) LSB[n] (0x0054 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.18.

6.3.13.19 Manageability EtherType Filter (METF) MSB[n] (0x0055 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.19.

6.3.13.20 ARP Response IPv4 Address LSB (0x005C)

For inner structure, see Section 6.3.11.20.

6.3.13.21 ARP Response IPv4 Address MSB (0x005D)

For inner structure, see Section 6.3.11.21.

6.3.13.22 IPv6 Address Bytes 0-1[n] (0x005E + 8*n, n=0...3)

For inner structure, see Section 6.3.11.22.

6.3.13.23 IPv6 Address Bytes 2-3[n] (0x005F + 8*n, n=0...3)

For inner structure, see Section 6.3.11.23.

6.3.13.24 IPv6 Address Bytes 4-5[n] (0x0060 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.24.

6.3.13.25 IPv6 Address Bytes 6-7[n] (0x0061 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.25.

6.3.13.26 IPv6 Address Bytes 8-9[n] (0x0062 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.26.

6.3.13.27 IPv6 Address Bytes 10-11[n] (0x0063 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.27.

6.3.13.28 IPv6 Address Bytes 12-13[n] (0x0064 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.28.

6.3.13.29 IPv6 Address Bytes 14-15[n] (0x0065 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.29.

6.3.13.30 Manageability Special Modifiers LSB (0x007E)

For inner structure, see Section 6.3.11.30.

6.3.13.31 Manageability Special Modifiers MSB (0x007F)

For inner structure, see Section 6.3.11.31.

6.3.14 PFA PT Configuration 3 Section

Pass-through filters of Port 3 PFA section header.

Table 6-20. PFA PT Configuration 3 Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type PT Module 3	6.3.14.1
0x0001	Section Length	6.3.14.2
0x0002 + 2*n, n=0...3	LAN IPv4 Address (LSB) MIPAF0	6.3.14.3
0x0003 + 2*n, n=0...3	LAN IPv4 Address (MSB) MIPAF0	6.3.14.4
0x000A + 2*n, n=0...15	LAN Flexible Filter Port	6.3.14.5
0x000B + 2*n, n=0...15	LAN Flexible Filter Port - Modifiers	6.3.14.6
0x002A + 1*n, n=0...7	LAN VLAN Filter	6.3.14.7
0x0032	LAN MANC Value LSB	6.3.14.8
0x0033	LAN MANC Value MSB	6.3.14.9
0x0034	Receive Enable 1 - LRXEN1	6.3.14.10
0x0035	Receive Enable 2 - LRXEN2	6.3.14.11
0x0036	LAN MNGONLY LSB	6.3.14.12
0x0037	LAN MNGONLY MSB	6.3.14.13
0x0038 + 4*n, n=0...6	Manageability Decision Filters LSB	6.3.14.14
0x0039 + 4*n, n=0...6	Manageability Decision Filters MSB	6.3.14.15
0x003A + 4*n, n=0...6	Manageability Extended Decision Filters LSB	6.3.14.16
0x003B + 4*n, n=0...6	Manageability Extended Decision Filters MSB	6.3.14.17
0x0054 + 2*n, n=0...3	Manageability EtherType Filter (METF) LSB	6.3.14.18
0x0055 + 2*n, n=0...3	Manageability EtherType Filter (METF) MSB	6.3.14.19
0x005C	ARP Response IPv4 Address LSB	6.3.14.20
0x005D	ARP Response IPv4 Address MSB	6.3.14.21
0x005E + 8*n, n=0...3	IPv6 Address Bytes 0-1	6.3.14.22
0x005F + 8*n, n=0...3	IPv6 Address Bytes 2-3	6.3.14.23
0x0060 + 8*n, n=0...3	IPv6 Address Bytes 4-5	6.3.14.24
0x0061 + 8*n, n=0...3	IPv6 Address Bytes 6-7	6.3.14.25
0x0062 + 8*n, n=0...3	IPv6 Address Bytes 8-9	6.3.14.26
0x0063 + 8*n, n=0...3	IPv6 Address Bytes 10-11	6.3.14.27
0x0064 + 8*n, n=0...3	IPv6 Address Bytes 12-13	6.3.14.28
0x0065 + 8*n, n=0...3	IPv6 Address Bytes 14-15	6.3.14.29
0x007E	Manageability Special Modifiers LSB	6.3.14.30
0x007F	Manageability Special Modifiers MSB	6.3.14.31

6.3.14.1 Sub Module Type PT Module 3 (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x10B	Valid values are: 0x10B = Pass-Through Control Words Structure 3

6.3.14.2 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, see Section 6.3.11.2.

6.3.14.3 LAN IPv4 Address (LSB) MIPAF0[n] (0x0002 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.3.

6.3.14.4 LAN IPv4 Address (MSB) MIPAF0[n] (0x0003 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.4.

6.3.14.5 LAN Flexible Filter Port[n] (0x000A + 2*n, n=0...15)

For inner structure, see Section 6.3.11.5.

6.3.14.6 LAN Flexible Filter Port - Modifiers[n] (0x000B + 2*n, n=0...15)

For inner structure, see Section 6.3.11.6.

6.3.14.7 LAN VLAN Filter[n] (0x002A + 1*n, n=0...7)

For inner structure, see Section 6.3.11.7.

6.3.14.8 LAN MANC Value LSB (0x0032)

For inner structure, see Section 6.3.11.8.

6.3.14.9 LAN MANC Value MSB (0x0033)

For inner structure, see Section 6.3.11.9.

6.3.14.10 Receive Enable 1 - LRXEN1 (0x0034)

For inner structure, see Section 6.3.11.10.

6.3.14.11 Receive Enable 2 - LRXEN2 (0x0035)

For inner structure, see Section 6.3.11.11.

6.3.14.12 LAN MNGONLY LSB (0x0036)

For inner structure, see Section 6.3.11.12.

6.3.14.13 LAN MNGONLY MSB (0x0037)

For inner structure, see Section 6.3.11.13.

6.3.14.14 Manageability Decision Filters LSB[n] (0x0038 + 4*n, n=0...6)

For inner structure, see Section 6.3.11.14.

6.3.14.15 Manageability Decision Filters MSB[n] (0x0039 + 4*n, n=0...6)

For inner structure, see Section 6.3.11.15.

6.3.14.16 Manageability Extended Decision Filters LSB[n] (0x003A + 4*n, n=0...6)

For inner structure, see Section 6.3.11.16.

6.3.14.17 Manageability Extended Decision Filters MSB[n] (0x003B + 4*n, n=0...6)

For inner structure, see Section 6.3.11.17.

6.3.14.18 Manageability EtherType Filter (METF) LSB[n] (0x0054 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.18.

6.3.14.19 Manageability EtherType Filter (METF) MSB[n] (0x0055 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.19.

6.3.14.20 ARP Response IPv4 Address LSB (0x005C)

For inner structure, see Section 6.3.11.20.

6.3.14.21 ARP Response IPv4 Address MSB (0x005D)

For inner structure, see Section 6.3.11.21.

6.3.14.22 IPv6 Address Bytes 0-1[n] (0x005E + 8*n, n=0...3)

For inner structure, see Section 6.3.11.22.

6.3.14.23 IPv6 Address Bytes 2-3[n] (0x005F + 8*n, n=0...3)

For inner structure, see Section 6.3.11.23.

6.3.14.24 IPv6 Address Bytes 4-5[n] (0x0060 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.24.

6.3.14.25 IPv6 Address Bytes 6-7[n] (0x0061 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.25.

6.3.14.26 IPv6 Address Bytes 8-9[n] (0x0062 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.26.

6.3.14.27 IPv6 Address Bytes 10-11[n] (0x0063 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.27.

6.3.14.28 IPv6 Address Bytes 12-13[n] (0x0064 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.28.

6.3.14.29 IPv6 Address Bytes 14-15[n] (0x0065 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.29.

6.3.14.30 Manageability Special Modifiers LSB (0x007E)

For inner structure, see Section 6.3.11.30.

6.3.14.31 Manageability Special Modifiers MSB (0x007F)

For inner structure, see Section 6.3.11.31.

6.3.15 PFA PT Configuration 4 Section

Pass-through filters of Port 4 PFA section header.

Table 6-21. PFA PT Configuration 4 Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type PT Module 4	6.3.15.1
0x0001	Section Length	6.3.15.2
0x0002 + 2*n, n=0...3	LAN IPv4 Address (LSB) MIPAF0	6.3.15.3
0x0003 + 2*n, n=0...3	LAN IPv4 Address (MSB) MIPAF0	6.3.15.4
0x000A + 2*n, n=0...15	LAN Flexible Filter Port	6.3.15.5
0x000B + 2*n, n=0...15	LAN Flexible Filter Port - Modifiers	6.3.15.6
0x002A + 1*n, n=0...7	LAN VLAN Filter	6.3.15.7
0x0032	LAN MANC Value LSB	6.3.15.8
0x0033	LAN MANC Value MSB	6.3.15.9
0x0034	Receive Enable 1 - LRXEN1	6.3.15.10
0x0035	Receive Enable 2 - LRXEN2	6.3.15.11
0x0036	LAN MNGONLY LSB	6.3.15.12
0x0037	LAN MNGONLY MSB	6.3.15.13
0x0038 + 4*n, n=0...6	Manageability Decision Filters LSB	6.3.15.14
0x0039 + 4*n, n=0...6	Manageability Decision Filters MSB	6.3.15.15
0x003A + 4*n, n=0...6	Manageability Extended Decision Filters LSB	6.3.15.16
0x003B + 4*n, n=0...6	Manageability Extended Decision Filters MSB	6.3.15.17
0x0054 + 2*n, n=0...3	Manageability EtherType Filter (METF) LSB	6.3.15.18
0x0055 + 2*n, n=0...3	Manageability EtherType Filter (METF) MSB	6.3.15.19
0x005C	ARP Response IPv4 Address LSB	6.3.15.20
0x005D	ARP Response IPv4 Address MSB	6.3.15.21
0x005E + 8*n, n=0...3	IPv6 Address Bytes 0-1	6.3.15.22
0x005F + 8*n, n=0...3	IPv6 Address Bytes 2-3	6.3.15.23
0x0060 + 8*n, n=0...3	IPv6 Address Bytes 4-5	6.3.15.24
0x0061 + 8*n, n=0...3	IPv6 Address Bytes 6-7	6.3.15.25
0x0062 + 8*n, n=0...3	IPv6 Address Bytes 8-9	6.3.15.26
0x0063 + 8*n, n=0...3	IPv6 Address Bytes 10-11	6.3.15.27
0x0064 + 8*n, n=0...3	IPv6 Address Bytes 12-13	6.3.15.28
0x0065 + 8*n, n=0...3	IPv6 Address Bytes 14-15	6.3.15.29
0x007E	Manageability Special Modifiers LSB	6.3.15.30
0x007F	Manageability Special Modifiers MSB	6.3.15.31

6.3.15.1 Sub Module Type PT Module 4 (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x114	Valid values are: 0x114 = Pass-through Control Word Structure 4

6.3.15.2 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, see Section 6.3.11.2.

6.3.15.3 LAN IPv4 Address (LSB) MIPAF0[n] (0x0002 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.3.

6.3.15.4 LAN IPv4 Address (MSB) MIPAF0[n] (0x0003 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.4.

6.3.15.5 LAN Flexible Filter Port[n] (0x000A + 2*n, n=0...15)

For inner structure, see Section 6.3.11.5.

6.3.15.6 LAN Flexible Filter Port - Modifiers[n] (0x000B + 2*n, n=0...15)

For inner structure, see Section 6.3.11.6.

6.3.15.7 LAN VLAN Filter[n] (0x002A + 1*n, n=0...7)

For inner structure, see Section 6.3.11.7.

6.3.15.8 LAN MANC Value LSB (0x0032)

For inner structure, see Section 6.3.11.8.

6.3.15.9 LAN MANC Value MSB (0x0033)

For inner structure, see Section 6.3.11.9.

6.3.15.10 Receive Enable 1 - LRXEN1 (0x0034)

For inner structure, see Section 6.3.11.10.

6.3.15.11 Receive Enable 2 - LRXEN2 (0x0035)

For inner structure, see Section 6.3.11.11.

6.3.15.12 LAN MNGONLY LSB (0x0036)

For inner structure, see Section 6.3.11.12.

6.3.15.13 LAN MNGONLY MSB (0x0037)

For inner structure, see Section 6.3.11.13.

6.3.15.14 Manageability Decision Filters LSB[n] (0x0038 + 4*n, n=0...6)

For inner structure, see Section 6.3.11.14.

6.3.15.15 Manageability Decision Filters MSB[n] (0x0039 + 4*n, n=0...6)

For inner structure, see Section 6.3.11.15.

6.3.15.16 Manageability Extended Decision Filters LSB[n] (0x003A + 4*n, n=0...6)

For inner structure, see Section 6.3.11.16.

6.3.15.17 Manageability Extended Decision Filters MSB[n] (0x003B + 4*n, n=0...6)

For inner structure, see Section 6.3.11.17.

6.3.15.18 Manageability EtherType Filter (METF) LSB[n] (0x0054 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.18.

6.3.15.19 Manageability EtherType Filter (METF) MSB[n] (0x0055 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.19.

6.3.15.20 ARP Response IPv4 Address LSB (0x005C)

For inner structure, see Section 6.3.11.20.

6.3.15.21 ARP Response IPv4 Address MSB (0x005D)

For inner structure, see Section 6.3.11.21.

6.3.15.22 IPv6 Address Bytes 0-1[n] (0x005E + 8*n, n=0...3)

For inner structure, see Section 6.3.11.22.

6.3.15.23 IPv6 Address Bytes 2-3[n] (0x005F + 8*n, n=0...3)

For inner structure, see Section 6.3.11.23.

6.3.15.24 IPv6 Address Bytes 4-5[n] (0x0060 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.24.

6.3.15.25 IPv6 Address Bytes 6-7[n] (0x0061 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.25.

6.3.15.26 IPv6 Address Bytes 8-9[n] (0x0062 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.26.

6.3.15.27 IPv6 Address Bytes 10-11[n] (0x0063 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.27.

6.3.15.28 IPv6 Address Bytes 12-13[n] (0x0064 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.28.

6.3.15.29 IPv6 Address Bytes 14-15[n] (0x0065 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.29.

6.3.15.30 Manageability Special Modifiers LSB (0x007E)

For inner structure, see Section 6.3.11.30.

6.3.15.31 Manageability Special Modifiers MSB (0x007F)

For inner structure, see Section 6.3.11.31.

6.3.16 PFA PT Configuration 5 Section

Pass-through filters of Port 5 PFA section header.

Table 6-22. PFA PT Configuration 5 Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type PT Module 5	6.3.16.1
0x0001	Section Length	6.3.16.2
0x0002 + 2*n, n=0...3	LAN IPv4 Address (LSB) MIPAF0	6.3.16.3
0x0003 + 2*n, n=0...3	LAN IPv4 Address (MSB) MIPAF0	6.3.16.4
0x000A + 2*n, n=0...15	LAN Flexible Filter Port	6.3.16.5
0x000B + 2*n, n=0...15	LAN Flexible Filter Port - Modifiers	6.3.16.6
0x002A + 1*n, n=0...7	LAN VLAN Filter	6.3.16.7
0x0032	LAN MANC Value LSB	6.3.16.8
0x0033	LAN MANC Value MSB	6.3.16.9
0x0034	Receive Enable 1 - LRXEN1	6.3.16.10
0x0035	Receive Enable 2 - LRXEN2	6.3.16.11
0x0036	LAN MNGONLY LSB	6.3.16.12
0x0037	LAN MNGONLY MSB	6.3.16.13
0x0038 + 4*n, n=0...6	Manageability Decision Filters LSB	6.3.16.14
0x0039 + 4*n, n=0...6	Manageability Decision Filters MSB	6.3.16.15
0x003A + 4*n, n=0...6	Manageability Extended Decision Filters LSB	6.3.16.16
0x003B + 4*n, n=0...6	Manageability Extended Decision Filters MSB	6.3.16.17
0x0054 + 2*n, n=0...3	Manageability EtherType Filter (METF) LSB	6.3.16.18
0x0055 + 2*n, n=0...3	Manageability EtherType Filter (METF) MSB	6.3.16.19
0x005C	ARP Response IPv4 Address LSB	6.3.16.20
0x005D	ARP Response IPv4 Address MSB	6.3.16.21
0x005E + 8*n, n=0...3	IPv6 Address Bytes 0-1	6.3.16.22
0x005F + 8*n, n=0...3	IPv6 Address Bytes 2-3	6.3.16.23
0x0060 + 8*n, n=0...3	IPv6 Address Bytes 4-5	6.3.16.24
0x0061 + 8*n, n=0...3	IPv6 Address Bytes 6-7	6.3.16.25
0x0062 + 8*n, n=0...3	IPv6 Address Bytes 8-9	6.3.16.26
0x0063 + 8*n, n=0...3	IPv6 Address Bytes 10-11	6.3.16.27
0x0064 + 8*n, n=0...3	IPv6 Address Bytes 12-13	6.3.16.28
0x0065 + 8*n, n=0...3	IPv6 Address Bytes 14-15	6.3.16.29
0x007E	Manageability Special Modifiers LSB	6.3.16.30
0x007F	Manageability Special Modifiers MSB	6.3.16.31

6.3.16.1 Sub Module Type PT Module 5 (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x115	Valid values are: 0x115 = Pass-through Control Word Structure 5

6.3.16.2 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, see Section 6.3.11.2.

6.3.16.3 LAN IPv4 Address (LSB) MIPAF0[n] (0x0002 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.3.

6.3.16.4 LAN IPv4 Address (MSB) MIPAF0[n] (0x0003 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.4.

6.3.16.5 LAN Flexible Filter Port[n] (0x000A + 2*n, n=0...15)

For inner structure, see Section 6.3.11.5.

6.3.16.6 LAN Flexible Filter Port - Modifiers[n] (0x000B + 2*n, n=0...15)

For inner structure, see Section 6.3.11.6.

6.3.16.7 LAN VLAN Filter[n] (0x002A + 1*n, n=0...7)

For inner structure, see Section 6.3.11.7.

6.3.16.8 LAN MANC Value LSB (0x0032)

For inner structure, see Section 6.3.11.8.

6.3.16.9 LAN MANC Value MSB (0x0033)

For inner structure, see Section 6.3.11.9.

6.3.16.10 Receive Enable 1 - LRXEN1 (0x0034)

For inner structure, see Section 6.3.11.10.

6.3.16.11 Receive Enable 2 - LRXEN2 (0x0035)

For inner structure, see Section 6.3.11.11.

6.3.16.12 LAN MNGONLY LSB (0x0036)

For inner structure, see Section 6.3.11.12.

6.3.16.13 LAN MNGONLY MSB (0x0037)

For inner structure, see Section 6.3.11.13.

6.3.16.14 Manageability Decision Filters LSB[n] (0x0038 + 4*n, n=0...6)

For inner structure, see Section 6.3.11.14.

6.3.16.15 Manageability Decision Filters MSB[n] (0x0039 + 4*n, n=0...6)

For inner structure, see Section 6.3.11.15.

6.3.16.16 Manageability Extended Decision Filters LSB[n] (0x003A + 4*n, n=0...6)

For inner structure, see Section 6.3.11.16.

6.3.16.17 Manageability Extended Decision Filters MSB[n] (0x003B + 4*n, n=0...6)

For inner structure, see Section 6.3.11.17.

6.3.16.18 Manageability EtherType Filter (METF) LSB[n] (0x0054 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.18.

6.3.16.19 Manageability EtherType Filter (METF) MSB[n] (0x0055 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.19.

6.3.16.20 ARP Response IPv4 Address LSB (0x005C)

For inner structure, see Section 6.3.11.20.

6.3.16.21 ARP Response IPv4 Address MSB (0x005D)

For inner structure, see Section 6.3.11.21.

6.3.16.22 IPv6 Address Bytes 0-1[n] (0x005E + 8*n, n=0...3)

For inner structure, see Section 6.3.11.22.

6.3.16.23 IPv6 Address Bytes 2-3[n] (0x005F + 8*n, n=0...3)

For inner structure, see Section 6.3.11.23.

6.3.16.24 IPv6 Address Bytes 4-5[n] (0x0060 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.24.

6.3.16.25 IPv6 Address Bytes 6-7[n] (0x0061 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.25.

6.3.16.26 IPv6 Address Bytes 8-9[n] (0x0062 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.26.

6.3.16.27 IPv6 Address Bytes 10-11[n] (0x0063 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.27.

6.3.16.28 IPv6 Address Bytes 12-13[n] (0x0064 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.28.

6.3.16.29 IPv6 Address Bytes 14-15[n] (0x0065 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.29.

6.3.16.30 Manageability Special Modifiers LSB (0x007E)

For inner structure, see Section 6.3.11.30.

6.3.16.31 Manageability Special Modifiers MSB (0x007F)

For inner structure, see Section 6.3.11.31.

6.3.17 PFA PT Configuration 6 Section

Pass-through filters of Port 6 PFA section header.

Table 6-23. PFA PT Configuration 6 Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type PT Module 6	6.3.17.1
0x0001	Section Length	6.3.17.2
0x0002 + 2*n, n=0...3	LAN IPv4 Address (LSB) MIPAF0	6.3.17.3
0x0003 + 2*n, n=0...3	LAN IPv4 Address (MSB) MIPAF0	6.3.17.4
0x000A + 2*n, n=0...15	LAN Flexible Filter Port	6.3.17.5
0x000B + 2*n, n=0...15	LAN Flexible Filter Port - Modifiers	6.3.17.6
0x002A + 1*n, n=0...7	LAN VLAN Filter	6.3.17.7
0x0032	LAN MANC Value LSB	6.3.17.8
0x0033	LAN MANC Value MSB	6.3.17.9
0x0034	Receive Enable 1 - LRXEN1	6.3.17.10
0x0035	Receive Enable 2 - LRXEN2	6.3.17.11
0x0036	LAN MNGONLY LSB	6.3.17.12
0x0037	LAN MNGONLY MSB	6.3.17.13
0x0038 + 4*n, n=0...6	Manageability Decision Filters LSB	6.3.17.14
0x0039 + 4*n, n=0...6	Manageability Decision Filters MSB	6.3.17.15
0x003A + 4*n, n=0...6	Manageability Extended Decision Filters LSB	6.3.17.16
0x003B + 4*n, n=0...6	Manageability Extended Decision Filters MSB	6.3.17.17
0x0054 + 2*n, n=0...3	Manageability EtherType Filter (METF) LSB	6.3.17.18
0x0055 + 2*n, n=0...3	Manageability EtherType Filter (METF) MSB	6.3.17.19
0x005C	ARP Response IPv4 Address LSB	6.3.17.20
0x005D	ARP Response IPv4 Address MSB	6.3.17.21
0x005E + 8*n, n=0...3	IPv6 Address Bytes 0-1	6.3.17.22
0x005F + 8*n, n=0...3	IPv6 Address Bytes 2-3	6.3.17.23
0x0060 + 8*n, n=0...3	IPv6 Address Bytes 4-5	6.3.17.24
0x0061 + 8*n, n=0...3	IPv6 Address Bytes 6-7	6.3.17.25
0x0062 + 8*n, n=0...3	IPv6 Address Bytes 8-9	6.3.17.26
0x0063 + 8*n, n=0...3	IPv6 Address Bytes 10-11	6.3.17.27
0x0064 + 8*n, n=0...3	IPv6 Address Bytes 12-13	6.3.17.28
0x0065 + 8*n, n=0...3	IPv6 Address Bytes 14-15	6.3.17.29
0x007E	Manageability Special Modifiers LSB	6.3.17.30
0x007F	Manageability Special Modifiers MSB	6.3.17.31

6.3.17.1 Sub Module Type PT Module 6 (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x116	Valid values are: 0x116 = Pass-through Control Word Structure 6

6.3.17.2 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, see Section 6.3.11.2.

6.3.17.3 LAN IPv4 Address (LSB) MIPAF0[n] (0x0002 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.3.

6.3.17.4 LAN IPv4 Address (MSB) MIPAF0[n] (0x0003 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.4.

6.3.17.5 LAN Flexible Filter Port[n] (0x000A + 2*n, n=0...15)

For inner structure, see Section 6.3.11.5.

6.3.17.6 LAN Flexible Filter Port - Modifiers[n] (0x000B + 2*n, n=0...15)

For inner structure, see Section 6.3.11.6.

6.3.17.7 LAN VLAN Filter[n] (0x002A + 1*n, n=0...7)

For inner structure, see Section 6.3.11.7.

6.3.17.8 LAN MANC Value LSB (0x0032)

For inner structure, see Section 6.3.11.8.

6.3.17.9 LAN MANC Value MSB (0x0033)

For inner structure, see Section 6.3.11.9.

6.3.17.10 Receive Enable 1 - LRXEN1 (0x0034)

For inner structure, see Section 6.3.11.10.

6.3.17.11 Receive Enable 2 - LRXEN2 (0x0035)

For inner structure, see Section 6.3.11.11.

6.3.17.12 LAN MNGONLY LSB (0x0036)

For inner structure, see Section 6.3.11.12.

6.3.17.13 LAN MNGONLY MSB (0x0037)

For inner structure, see Section 6.3.11.13.

6.3.17.14 Manageability Decision Filters LSB[n] (0x0038 + 4*n, n=0...6)

For inner structure, see Section 6.3.11.14.

6.3.17.15 Manageability Decision Filters MSB[n] (0x0039 + 4*n, n=0...6)

For inner structure, see Section 6.3.11.15.

6.3.17.16 Manageability Extended Decision Filters LSB[n] (0x003A + 4*n, n=0...6)

For inner structure, see Section 6.3.11.16.

6.3.17.17 Manageability Extended Decision Filters MSB[n] (0x003B + 4*n, n=0...6)

For inner structure, see Section 6.3.11.17.

6.3.17.18 Manageability EtherType Filter (METF) LSB[n] (0x0054 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.18.

6.3.17.19 Manageability EtherType Filter (METF) MSB[n] (0x0055 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.19.

6.3.17.20 ARP Response IPv4 Address LSB (0x005C)

For inner structure, see Section 6.3.11.20.

6.3.17.21 ARP Response IPv4 Address MSB (0x005D)

For inner structure, see Section 6.3.11.21.

6.3.17.22 IPv6 Address Bytes 0-1[n] (0x005E + 8*n, n=0...3)

For inner structure, see Section 6.3.11.22.

6.3.17.23 IPv6 Address Bytes 2-3[n] (0x005F + 8*n, n=0...3)

For inner structure, see Section 6.3.11.23.

6.3.17.24 IPv6 Address Bytes 4-5[n] (0x0060 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.24.

6.3.17.25 IPv6 Address Bytes 6-7[n] (0x0061 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.25.

6.3.17.26 IPv6 Address Bytes 8-9[n] (0x0062 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.26.

6.3.17.27 IPv6 Address Bytes 10-11[n] (0x0063 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.27.

6.3.17.28 IPv6 Address Bytes 12-13[n] (0x0064 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.28.

6.3.17.29 IPv6 Address Bytes 14-15[n] (0x0065 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.29.

6.3.17.30 Manageability Special Modifiers LSB (0x007E)

For inner structure, see Section 6.3.11.30.

6.3.17.31 Manageability Special Modifiers MSB (0x007F)

For inner structure, see Section 6.3.11.31.

6.3.18 PFA PT Configuration 7 Section

Pass-through filters of Port 7 PFA section header.

Table 6-24. PFA PT Configuration 7 Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type PT Module 7	6.3.18.1
0x0001	Section Length	6.3.18.2
0x0002 + 2*n, n=0...3	LAN IPv4 Address (LSB) MIPAF0	6.3.18.3
0x0003 + 2*n, n=0...3	LAN IPv4 Address (MSB) MIPAF0	6.3.18.4
0x000A + 2*n, n=0...15	LAN Flexible Filter Port	6.3.18.5
0x000B + 2*n, n=0...15	LAN Flexible Filter Port - Modifiers	6.3.18.6
0x002A + 1*n, n=0...7	LAN VLAN Filter	6.3.18.7
0x0032	LAN MANC Value LSB	6.3.18.8
0x0033	LAN MANC Value MSB	6.3.18.9
0x0034	Receive Enable 1 - LRXEN1	6.3.18.10
0x0035	Receive Enable 2 - LRXEN2	6.3.18.11
0x0036	LAN MNGONLY LSB	6.3.18.12
0x0037	LAN MNGONLY MSB	6.3.18.13
0x0038 + 4*n, n=0...6	Manageability Decision Filters LSB	6.3.18.14
0x0039 + 4*n, n=0...6	Manageability Decision Filters MSB	6.3.18.15
0x003A + 4*n, n=0...6	Manageability Extended Decision Filters LSB	6.3.18.16
0x003B + 4*n, n=0...6	Manageability Extended Decision Filters MSB	6.3.18.17
0x0054 + 2*n, n=0...3	Manageability EtherType Filter (METF) LSB	6.3.18.18
0x0055 + 2*n, n=0...3	Manageability EtherType Filter (METF) MSB	6.3.18.19
0x005C	ARP Response IPv4 Address LSB	6.3.18.20
0x005D	ARP Response IPv4 Address MSB	6.3.18.21
0x005E + 8*n, n=0...3	IPv6 Address Bytes 0-1	6.3.18.22
0x005F + 8*n, n=0...3	IPv6 Address Bytes 2-3	6.3.18.23
0x0060 + 8*n, n=0...3	IPv6 Address Bytes 4-5	6.3.18.24
0x0061 + 8*n, n=0...3	IPv6 Address Bytes 6-7	6.3.18.25
0x0062 + 8*n, n=0...3	IPv6 Address Bytes 8-9	6.3.18.26
0x0063 + 8*n, n=0...3	IPv6 Address Bytes 10-11	6.3.18.27
0x0064 + 8*n, n=0...3	IPv6 Address Bytes 12-13	6.3.18.28
0x0065 + 8*n, n=0...3	IPv6 Address Bytes 14-15	6.3.18.29
0x007E	Manageability Special Modifiers LSB	6.3.18.30
0x007F	Manageability Special Modifiers MSB	6.3.18.31

6.3.18.1 Sub Module Type PT Module 7 (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x117	Valid values are: 0x117 = Pass-through Control Word Structure 7

6.3.18.2 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, see Section 6.3.11.2.

6.3.18.3 LAN IPv4 Address (LSB) MIPAF0[n] (0x0002 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.3.

6.3.18.4 LAN IPv4 Address (MSB) MIPAF0[n] (0x0003 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.4.

6.3.18.5 LAN Flexible Filter Port[n] (0x000A + 2*n, n=0...15)

For inner structure, see Section 6.3.11.5.

6.3.18.6 LAN Flexible Filter Port - Modifiers[n] (0x000B + 2*n, n=0...15)

For inner structure, see Section 6.3.11.6.

6.3.18.7 LAN VLAN Filter[n] (0x002A + 1*n, n=0...7)

For inner structure, see Section 6.3.11.7.

6.3.18.8 LAN MANC Value LSB (0x0032)

For inner structure, see Section 6.3.11.8.

6.3.18.9 LAN MANC Value MSB (0x0033)

For inner structure, see Section 6.3.11.9.

6.3.18.10 Receive Enable 1 - LRXEN1 (0x0034)

For inner structure, see Section 6.3.11.10.

6.3.18.11 Receive Enable 2 - LRXEN2 (0x0035)

For inner structure, see Section 6.3.11.11.

6.3.18.12 LAN MNGONLY LSB (0x0036)

For inner structure, see Section 6.3.11.12.

6.3.18.13 LAN MNGONLY MSB (0x0037)

For inner structure, see Section 6.3.11.13.

6.3.18.14 Manageability Decision Filters LSB[n] (0x0038 + 4*n, n=0...6)

For inner structure, see Section 6.3.11.14.

6.3.18.15 Manageability Decision Filters MSB[n] (0x0039 + 4*n, n=0...6)

For inner structure, see Section 6.3.11.15.

6.3.18.16 Manageability Extended Decision Filters LSB[n] (0x003A + 4*n, n=0...6)

For inner structure, see Section 6.3.11.16.

6.3.18.17 Manageability Extended Decision Filters MSB[n] (0x003B + 4*n, n=0...6)

For inner structure, see Section 6.3.11.17.

6.3.18.18 Manageability EtherType Filter (METF) LSB[n] (0x0054 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.18.

6.3.18.19 Manageability EtherType Filter (METF) MSB[n] (0x0055 + 2*n, n=0...3)

For inner structure, see Section 6.3.11.19.

6.3.18.20 ARP Response IPv4 Address LSB (0x005C)

For inner structure, see Section 6.3.11.20.

6.3.18.21 ARP Response IPv4 Address MSB (0x005D)

For inner structure, see Section 6.3.11.21.

6.3.18.22 IPv6 Address Bytes 0-1[n] (0x005E + 8*n, n=0...3)

For inner structure, see Section 6.3.11.22.

6.3.18.23 IPv6 Address Bytes 2-3[n] (0x005F + 8*n, n=0...3)

For inner structure, see Section 6.3.11.23.

6.3.18.24 IPv6 Address Bytes 4-5[n] (0x0060 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.24.

6.3.18.25 IPv6 Address Bytes 6-7[n] (0x0061 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.25.

6.3.18.26 IPv6 Address Bytes 8-9[n] (0x0062 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.26.

6.3.18.27 IPv6 Address Bytes 10-11[n] (0x0063 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.27.

6.3.18.28 IPv6 Address Bytes 12-13[n] (0x0064 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.28.

6.3.18.29 IPv6 Address Bytes 14-15[n] (0x0065 + 8*n, n=0...3)

For inner structure, see Section 6.3.11.29.

6.3.18.30 Manageability Special Modifiers LSB (0x007E)

For inner structure, see Section 6.3.11.30.

6.3.18.31 Manageability Special Modifiers MSB (0x007F)

For inner structure, see Section 6.3.11.31.

6.3.19 Original EETrack ID Section

Original EETrack ID PFA section.

Table 6-25. Original EETrack ID Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type - Original EETrackID	6.3.19.1
0x0001	Length	6.3.19.2
0x0002	Original EETrackID MSB	6.3.19.3
0x0003	Original EETrackID LSB	6.3.19.4

6.3.19.1 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

6.3.19.2 Length (0x0001)

Bit(s)	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

6.3.19.3 Original EETrackID MSB (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:0	EETrackID - MSB	0x0	

6.3.19.4 Original EETrackID LSB (0x0003)

Bit(s)	Field Name	Default NVM Value	Description
15:0	EETrackID - MLB	0x0	

6.3.20 IBA Capabilities Module Section

Table 6-26. IBA Capabilities Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type - IBA Capabilities	6.3.20.1
0x0001	Length	6.3.20.2
0x0002	IBA Capabilities	6.3.20.3

6.3.20.1 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

6.3.20.2 Length (0x0001)

Bit(s)	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

6.3.20.3 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.

Bit(s)	Field Name	Default NVM Value	Description
15:14	Signature	01b	Signature Must be set to 01b to indicate that this word has been programmed by the agent or other configuration software.
13:6	Reserved	0x0	Reserved. Must be 0.
5	Reserved	0b	FCoE boot code is present if set to 1b. 0b = Not Present 1b = Present
4	iSCSI Boot	0b	iSCSI is present if set to 1b. 0b = Not Present 1b = Present
3	EFI EBD Driver	0b	EFI UNDI driver is present if set to 1b. 0b = Not Present 1b = Present
2	RPL	0b	RPL module is present if set to 1b. Reserved bit for devices. 0b = Not Present 1b = Present

Bit(s)	Field Name	Default NVM Value	Description
1	PXE/UNDI Driver	1b	PXE UNDI driver is present if set to 1b. 0b = Not Present 1b = Present
0	PXE Base Code	1b	PXE Base Code is present if set to 1b. 0b = Not Present 1b = Present

6.3.21 PXE Setup Options Module Section

Table 6-27. PXE Setup Options Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type - PXE Setup	6.3.21.1
0x0001	Length	6.3.21.2
0x0002 + 1*n, n=0...7	Setup Options PCI Function	6.3.21.3

6.3.21.1 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

6.3.21.2 Length (0x0001)

Bit(s)	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

6.3.21.3 Setup Options PCI Function[n] (0x0002 + 1*n, n=0...7)

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.

Bit(s)	Field Name	Default NVM Value	Description
15:13	Reserved	0x0	Reserved. Must be 0x0.
12:10	FSD	000b	Bits 12-10 control forcing speed and duplex during driver operation. 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.
9	Reserved	0b	Reserved to legacy OS Wakeup Support (for 82559-based adapters only), which is not supported in the E810.
8	DSM	1b	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	0b	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.
5	iSCSI Boot Disabled	0b	When this bit is set and adapter port is neither iSCSI primary nor secondary, setup code must not be loaded. Otherwise iSCSI banner and Setup menu should be accessible as in current design. 0b = Enabled 1b = Disabled This bit must be changed at factory level and not be altered by any end-customer tools. For regular NIC and LOM design, this bit should be always cleared in the NVM image. Otherwise, iSCSI setup will not be accessible for the user.
4:3	DBS	00b	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. 00b = Network boot, then local boot (default) 01b = Local boot, then network boot 10b = Network boot only 11b = Local boot only
2:0	PS	000b	Protocol Select These bits select the active boot protocol. 000b = PXE (default value) 001b = RPL (only if RPL is in the Flash) 010b = iSCSI Boot primary port (only if iSCSI Boot is using this adapter) 011b = iSCSI Boot secondary port (only if iSCSI Boot is using this adapter). 100b = FCoE All other values are reserved. Only the default value of 000b should be initially programmed into the adapter. Other values should only be set by configuration utilities.

6.3.22 PXE Configuration Customization Options Module Section

Table 6-28. PXE Configuration Customization Options Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type - PXE Configuration Customization Options	6.3.22.1
0x0001	Length	6.3.22.2
0x0002 + 1*n, n=0...7	Configuration Customization Options PCI Function	6.3.22.3

6.3.22.1 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

6.3.22.2 Length (0x0001)

Bit(s)	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

6.3.22.3 Configuration Customization Options PCI Function[n] (0x0002 + 1*n, n=0...7)

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.

Bit(s)	Field Name	Default NVM Value	Description
15:14	Signature	01b	Signature Must be set to 01b to indicate that this word has been programmed by the agent or other configuration software.
13:12	Reserved	00b	Reserved. Must be 0b.

Bit(s)	Field Name	Default NVM Value	Description
11	Continuous Retry	0b	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. 0b = Disable (default) 1b = Enable
10:8	Operating mode	000b	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.
7:6	Reserved	00b	Reserved. Must be 00b
5	Disable Flash Update	0b	Disable Flash Update If this bit is set to 1b, the user is not allowed to update the flash image using PROSet. 0b = Enable Flash Update (default) 1b = Disable Flash Update
4	Disable Legacy OS Wakeup Menu	0b	Disable Legacy Wakeup Support If this bit is set to 1b, the user is not allowed to change the Legacy OS Wakeup Support menu option. 0b = Enable Legacy Wakeup Support (default) 1b = Disable Legacy Wakeup Support Default value is 0b.
3	Disable Boot Selection Menu	0b	Disable Boot Selection If this bit is set to 1b, the user is not allowed to change the boot order menu option. 0b = Enable (default) 1b = Disable
2	Disable Protocol Selection Menu	0b	Disable Protocol Select If set to 1b, the user is not allowed to change the boot protocol. 0b = Enable (default) 1b = Disable

Bit(s)	Field Name	Default NVM Value	Description
1	Disable Title Message Display	0b	Disable Title Message If this bit is set to 1b, 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. 0b = Enable (default) 1b = Disable
0	Setup Menu	0b	Disable Setup Menu If this bit is set to 1b, the user is not allowed to invoke the setup menu by pressing Control-S. In this case, the NVM can only be changed via an external program. 0b = Enable (default) 1b = Disable

6.3.23 PXE Version Module Section

Table 6-29. PXE Version Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type - PXE Version	6.3.23.1
0x0001	Length	6.3.23.2
0x0002	PXE Version	6.3.23.3

6.3.23.1 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

6.3.23.2 Length (0x0001)

Bit(s)	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

6.3.23.3 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.

Bit(s)	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.

6.3.24 VLAN Module Section

Table 6-30. VLAN Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type - VLAN	6.3.24.1
0x0001	Length	6.3.24.2
0x0002	VLAN Block Signature	6.3.24.3
0x0003	Structure Version and Size	6.3.24.4
0x0004	Port 0 VLAN Tag	6.3.24.5
0x0005	Port 1 VLAN Tag	6.3.24.6
0x0006	Port 2 VLAN Tag	6.3.24.7
0x0007	Port 3 VLAN Tag	6.3.24.8
0x0008	Port 4 VLAN Tag	6.3.24.9
0x0009	Port 5 VLAN Tag	6.3.24.10
0x000A	Port 6 VLAN Tag	6.3.24.11
0x000B	Port 7 VLAN Tag	6.3.24.12

6.3.24.1 Sub Module Type - VLAN (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x37	Valid values are: 0x37 = VLAN Configuration

6.3.24.2 Length (0x0001)

Bit(s)	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 7 VLAN Tag

6.3.24.3 VLAN Block Signature (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:0	VLAN Block Signature	0x4C56	'V'=0x56, 'L' = 0x4C

6.3.24.4 Structure Version and Size (0x0003)

Bit(s)	Field Name	Default NVM Value	Description
15:8	Structure Version and Size	0x0C	Total byte size of the configuration block. 0x06 = 1-port adapter 0x08 = 2-port adapter 0x0C = 4-port adapter (default)
7:0	Structure Version	0x01	The version of this structure. Should be set to 0x01.

6.3.24.5 Port 0 VLAN Tag (0x0004)

Bit(s)	Field Name	Default NVM Value	Description
15:13	VLAN Priority	0x0	The value of VLAN priority (0-7).
12	Reserved	0b	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.

6.3.24.6 Port 1 VLAN Tag (0x0005)

For inner structure, see Section 6.3.24.5.

6.3.24.7 Port 2 VLAN Tag (0x0006)

For inner structure, see Section 6.3.24.5.

6.3.24.8 Port 3 VLAN Tag (0x0007)

For inner structure, see Section 6.3.24.5.

6.3.24.9 Port 4 VLAN Tag (0x0008)

For inner structure, see Section 6.3.24.5.

6.3.24.10 Port 5 VLAN Tag (0x0009)

For inner structure, see Section 6.3.24.5.

6.3.24.11 Port 6 VLAN Tag (0x000A)

For inner structure, see Section 6.3.24.5.

6.3.24.12 Port 7 VLAN Tag (0x000B)

For inner structure, see Section 6.3.24.5.

6.3.25 Boot Configuration Block Section

Table 6-31. Boot Configuration Block Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type	6.3.25.1
0x0001	Length	6.3.25.2
0x0002	Combo Image Version High	6.3.25.3
0x0003	Combo Image Version Low	6.3.25.4

6.3.25.1 Sub Module Type (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x132	Valid values are: 0x132 = CIVD

6.3.25.2 Length (0x0001)

Bit(s)	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

6.3.25.3 Combo Image Version High (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:8	Major	0x0	
7:0	Build	0x0	

6.3.25.4 Combo Image Version Low (0x0003)

Bit(s)	Field Name	Default NVM Value	Description
15:8	Build	0x0	
7:0	Patch	0x0	

6.3.26 PBA Header Section

Table 6-32. PBA Header Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type - PBA	6.3.26.1
0x0001	Length	6.3.26.2

6.3.26.1 Sub Module Type - PBA (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x16	Valid values are: 0x16 = PBA block

6.3.26.2 Length (0x0001)

Bit(s)	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

6.3.27 PBA Block Section

The PBA block contains the complete PBA number, including the dash and the first digit of the 3-digit suffix.

Table 6-33. PBA Block Section Summary Table

Word Offset	Description	Section Reference
0x0000	PBA Section Length	6.3.27.1
0x0001	Word1	6.3.27.2
0x0002	Word2	6.3.27.3
0x0003	Word3	6.3.27.4
0x0004	Word4	6.3.27.5
0x0005	Word5	6.3.27.6

6.3.27.1 PBA Section Length (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PBA Section Length Field	0x6	Length in words of the PBA Block.

6.3.27.2 Word1 (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Word1 Field		PBA Number stored in hexadecimal ASCII values.

6.3.27.3 Word2 (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Word2 Field		PBA Number stored in hexadecimal ASCII values.

6.3.27.4 Word3 (0x0003)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Word3 Field		PBA Number stored in hexadecimal ASCII values.

6.3.27.5 Word4 (0x0004)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Word4 Field		PBA Number stored in hexadecimal ASCII values.

6.3.27.6 Word5 (0x0005)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Word5 Field		PBA Number stored in hexadecimal ASCII values.

6.3.28 PCIR Registers PFA Auto-Load Module Section

Table 6-34. PCIR Registers PFA Auto-Load Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	PCIR Registers Auto-Load Type	6.3.28.1
0x0001	Module Length	6.3.28.2
0x0002 - 0x0014	NVM contents for PFINT_ALLOC_PCI	6.3.28.3
0x0015 - 0x0027	NVM contents for PFPCI_SUBSYSID	6.3.28.4
0x0028 - 0x003A	NVM contents for PF_VT_PFALLOC_HIF	6.3.28.5
0x003B - 0x004D	NVM contents for PFPCI_DEVID	6.3.28.6
0x004E - 0x0052	NVM contents for GLPCI_CAPCTRL	6.3.28.7
0x0053 - 0x0054	NVM contents for GLPCI_CAPSUP	6.3.28.8
0x0055 - 0x0056	NVM contents for GLPCI_LINKCAP	6.3.28.9
0x0057 - 0x005A	NVM contents for GLPCI_VENDORID	6.3.28.10
0x005B - 0x005E	NVM contents for GLPCI_SUBVENID	6.3.28.11
0x005F - 0x0071	NVM contents for PFPCI_CNF	6.3.28.12
0x0072 - 0x0075	Reserved	6.3.28.13
0x0075 - 0x0088	NVM contents for PF_VT_PFALLOC_PCIE	6.3.28.14

6.3.28.1 PCIR Registers Auto-Load Type (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x113	Valid values are: 0x113 = PCIR Registers Auto-load

6.3.28.2 Module Length (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 2 First Section -> Word: PCIR Registers PFA Auto-Load Module -> PCIR Registers Auto-Load Type Last Section -> Word: PCIR Registers PFA Auto-Load Module -> Starting Address Low at PF_VT_PFALLOC_PCIE, for PF[0]

6.3.28.3 PFINT_ALLOC_PCI (0x0002 - 0x0014)

6.3.28.3.1 Starting Address Low at PFINT_ALLOC_PCI (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFINT_ALLOC_PCI, for PF[0]	0x9D800	
3:0	Type	0x2	

6.3.28.3.2 Starting Address High at PFINT_ALLOC_PCI (0x0003)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFINT_ALLOC_PCI, for PF[0]		

6.3.28.3.3 Attributes at PFINT_ALLOC_PCI (0x0004)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.28.3.4 Data Low of PFINT_ALLOC_PCI[PF] (0x0005 + 2*PF, PF=0...7)

6.3.28.3.5 Data High of PFINT_ALLOC_PCI[PF] (0x0006 + 2*PF, PF=0...7)

6.3.28.4 PFPCI_SUBSYSID (0x0015 - 0x0027)

6.3.28.4.1 Starting Address Low at PFPCI_SUBSYSID (0x0015)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFPCI_SUBSYSID, for PF[0]	0x9D880	
3:0	Type	0x2	

6.3.28.4.2 Starting Address High at PFPCI_SUBSYSID (0x0016)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFPCI_SUBSYSID, for PF[0]		

6.3.28.4.3 Attributes at PFPCI_FUNC2[PF] (0x0017)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.28.4.4 Data Low of PFPCI_SUBSYSID[PF] (0x0018 + 2*PF, PF=0...7)

6.3.28.4.5 Data High of PFPCI_SUBSYSID[PF] (0x0019 + 2*PF, PF=0...7)

6.3.28.5 PF_VT_PFALLOC_HIF (0x0028 - 0x003A)

6.3.28.5.1 Starting Address Low at PF_VT_PFALLOC_HIF (0x0028)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PF_VT_PFALLOC_HIF, for PF[0]	0x9DD80	
3:0	Type	0x2	

6.3.28.5.2 Starting Address High at PF_VT_PFALLOC_HIF (0x0029)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PF_VT_PFALLOC_HIF, for PF[0]		

6.3.28.5.3 Attributes at PF_VT_PFALLOC_HIF (0x002A)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.28.5.4 Data Low of PF_VT_PFALLOC_HIF[PF] (0x002B + 2*PF, PF=0...7)

6.3.28.5.5 Data High of PF_VT_PFALLOC_HIF[PF] (0x002C + 2*PF, PF=0...7)

6.3.28.6 PFPCI_DEVID (0x003B - 0x004D)

6.3.28.6.1 Starting Address Low at PFPCI_DEVID (0x003B)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFPCI_DEVID, for PF[0]	0x9DE00	
3:0	Type	0x2	

6.3.28.6.2 Starting Address High at PFPCI_DEVID (0x003C)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFPCI_DEVID, for PF[0]		

6.3.28.6.3 Attributes at PFPCI_DEVID (0x003D)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.28.6.4 Data Low of PFPCI_DEVID[PF] (0x003E + 2*PF, PF=0...7)

6.3.28.6.5 Data High of PFPCI_DEVID[PF] (0x003F + 2*PF, PF=0...7)

6.3.28.7 GLPCI_CAPCTRL (0x004E - 0x0052)

6.3.28.7.1 Starting Address Low at GLPCI_CAPCTRL (0x004E)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLPCI_CAPCTRL	0x9DE88	
3:0	Type	0X2	

6.3.28.7.2 Starting Address High at GLPCI_CAPCTRL (0x004F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLPCI_CAPCTRL		

6.3.28.7.3 Attributes at GLPCI_CAPCTRL (0x0050)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x3	
4:3	Skip	00b	
2:0	Width	000b	

6.3.28.7.4 Data Low of GLPCI_CAPCTRL (0x0051)

6.3.28.7.5 Data High of GLPCI_CAPCTRL (0x0052)

6.3.28.8 GLPCI_CAPSUP (0x0053 - 0x0054)

6.3.28.8.1 Data Low of GLPCI_CAPSUP (0x0053)

6.3.28.8.2 Data High of GLPCI_CAPSUP (0x0054)

6.3.28.9 GLPCI_LINKCAP (0x0055 - 0x0056)

6.3.28.9.1 Data Low of GLPCI_LINKCAP (0x0055)

6.3.28.9.2 Data High of GLPCI_LINKCAP (0x0056)

6.3.28.10 GLPCI_VENDORID (0x0057 - 0x005A)

6.3.28.10.1 Address Low at GLPCI_VENDORID (0x0057)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLPCI_VENDORID	0x9DEC8	
3:0	Type	0x1	

6.3.28.10.2 Address High at GLPCI_VENDORID (0x0058)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLPCI_VENDORID		

6.3.28.10.3 Data Low of GLPCI_VENDORID (0x0059)

6.3.28.10.4 Data High of GLPCI_VENDORID (0x005A)

6.3.28.11 GLPCI_SUBVENID (0x005B - 0x005E)

6.3.28.11.1 Address Low at GLPCI_SUBVENID (0x005B)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLPCI_SUBVENID	0x9DEE8	
3:0	Type	0x1	

6.3.28.11.2 Address High at GLPCI_SUBVENID (0x005C)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLPCI_SUBVENID		

6.3.28.11.3 Data Low of GLPCI_SUBVENID (0x005D)

6.3.28.11.4 Data High of GLPCI_SUBVENID (0x005E)

6.3.28.12 PFPCI_CNF (0x005F - 0x0071)

6.3.28.12.1 Starting Address Low at PFPCI_CNF (0x005F)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFPCI_CNF, for PF[0]	0x9DF00	
3:0	Type	0x2	

6.3.28.12.2 Starting Address High at PFPCI_CNF (0x0060)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFPCI_CNF, for PF[0]		

6.3.28.12.3 Attributes at PFPCI_CNF (0x0061)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.28.12.4 Data Low of PFPCI_CNF[PF] (0x0062 + 2*PF, PF=0...7)

6.3.28.12.5 Data High of PFPCI_CNF[PF] (0x0063 + 2*PF, PF=0...7)

6.3.28.13 Reserved (0x0072 - 0x0075)

6.3.28.14 PF_VT_PFALLOC_PCIE (0x0076 - 0x0088)

6.3.28.14.1 Starting Address Low at PF_VT_PFALLOC_PCIE (0x0076)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PF_VT_PFALLOC_PCIE, for PF[0]	0xBE080	
3:0	Type	0x2	

6.3.28.14.2 Starting Address High at PF_VT_PFALLOC_PCIE (0x0077)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PF_VT_PFALLOC_PCIE, for PF[0]		

6.3.28.14.3 Attributes at PF_VT_PFALLOC_PCIE (0x0078)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.28.14.4 Data Low of PF_VT_PFALLOC_PCIE[PF] (0x0079 + 2*PF, PF=0...7)

6.3.28.14.5 Data High of PF_VT_PFALLOC_PCIE[PF] (0x007A + 2*PF, PF=0...7)

6.3.29 POR Registers PFA Auto-Load Module Section

Register addresses and values loaded at Power-On Reset (POR).

Table 6-35. POR Registers PFA Auto-Load Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	POR Registers Auto-Load Type	6.3.29.1
0x0001	Module Length	6.3.29.2
0x0002 - 0x0012	NVM contents for GLGEN_GPIO_CTL	6.3.29.3
0x0013 - 0x0025	NVM contents for PFPCI_FUNC	6.3.29.4
0x0026 - 0x0038	NVM contents for PFPM_WUC	6.3.29.5
0x0039 - 0x003C	NVM contents for GLPCI_LBARCTRL	6.3.29.6
0x003D - 0x0040	NVM contents for GLPCI_CNF	6.3.29.7
0x0041 - 0x0044	NVM contents for GL_MNG_HWARB_CTRL	6.3.29.8
0x0045 - 0x0057	Reserved	6.3.29.9
0x0058 - 0x006A	NVM contents for PFPM_APM	6.3.29.10
0x006B - 0x007D	NVM contents for PRTGEN_CNF	6.3.29.11
0x007E - 0x008D	Reserved	6.3.29.12
0x008E - 0x009D	NVM contents for PRTGEN_CNF2	6.3.29.13
0x009E - 0x00A8	Reserved	6.3.29.14
0x00A9 - 0x00AC	NVM contents for GL_PWR_MODE_CTL	6.3.29.15

6.3.29.1 POR Registers Auto-Load Type (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x118	Valid values are: 0x118 =POR Registers Auto-load

6.3.29.2 Module Length (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 2 First Section -> Word: POR Registers PFA Auto-Load Module -> POR Registers Auto-Load Type Last Section -> Word: POR Registers PFA Auto-Load Module -> Address Low at GL_PWR_MODE_CTL

6.3.29.3 GLGEN_GPIO_CTL (0x0002 - 0x0012)

6.3.29.3.1 Starting Address Low at GLGEN_GPIO_CTL (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLGEN_GPIO_CTL	0x880C8	
3:0	Type	0x2	

6.3.29.3.2 Starting Address High at GLGEN_GPIO_CTL (0x0003)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLGEN_GPIO_CTL		

6.3.29.3.3 Attributes at GLGEN_GPIO_CTL (0x0004)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x7	
4:3	Skip	00b	
2:0	Width	000b	

6.3.29.3.4 Data Low of GLGEN_GPIO_CTL[n] (0x0005 + 2*n, n=0...6)

6.3.29.3.5 Data High of GLGEN_GPIO_CTL[n] (0x0006 + 2*n, n=0...6)

6.3.29.4 PFPCI_FUNC (0x0013 - 0x0025)

6.3.29.4.1 Starting Address Low at PFPCI_FUNC (0x0013)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFPCI_FUNC, for PF[0]	0x9D980	
3:0	Type	0x2	

6.3.29.4.2 Starting Address High at PFPCI_FUNC (0x0014)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFPCI_FUNC, for PF[0]		

6.3.29.4.3 Attributes at PFPCI_FUNC (0x0015)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.29.4.4 Data Low of PFPCI_FUNC[PF] (0x0016 + 2*PF, PF=0...7)

6.3.29.4.5 Data High of PFPCI_FUNC[PF] (0x0017 + 2*PF, PF=0...7)

6.3.29.5 PFPM_WUC (0x0026 - 0x0038)

6.3.29.5.1 Starting Address Low at PFPM_WUC 0x0026)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFPM_WUC, for PF[0]	0x9DC80	
3:0	Type	0x2	

6.3.29.5.2 Starting Address High at PFPM_WUC (0x0027)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFPM_WUC, for PF[0]		

6.3.29.5.3 Attributes at PFPM_WUC - 0x0028

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.29.5.4 Data Low of PFPM_WUC[PF] (0x0029 + 2*PF, PF=0...7)

6.3.29.5.5 Data High of PFPM_WUC[PF] (0x002A + 2*PF, PF=0...7)

6.3.29.6 GLPCI_LBARCTRL (0x0039 - 0x003C)

6.3.29.6.1 Address Low at GLPCI_LBARCTRL (0x0039)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLPCI_LBARCTRL	0x9DE74	
3:0	Type	0x1	

6.3.29.6.2 Address High at GLPCI_LBARCTRL (0x003A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLPCI_LBARCTRL		

6.3.29.6.3 Data Low of GLPCI_LBARCTRL (0x003B)

6.3.29.6.4 Data High of GLPCI_LBARCTRL (0x003C)

6.3.29.7 GLPCI_CNF (0x003D - 0x0040)

6.3.29.7.1 Address Low at GLPCI_CNF (0x003D)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLPCI_CNF	0x9DEA0	
3:0	Type	0x1	

6.3.29.7.2 Address High at GLPCI_CNF (0x003E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLPCI_CNF		

6.3.29.7.3 Data Low of GLPCI_CNF (0x003F)

6.3.29.7.4 Data High of GLPCI_CNF (0x0040)

6.3.29.8 GL_MNG_HWARB_CTRL (0x0041 - 0x0044)

6.3.29.8.1 Address Low at GL_MNG_HWARB_CTRL (0x0041)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GL_MNG_HWARB_CTRL	0xB6130	
3:0	Type	0x1	

6.3.29.8.2 Address High at GL_MNG_HWARB_CTRL (0x0042)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GL_MNG_HWARB_CTRL		

6.3.29.8.3 Data Low of GL_MNG_HWARB_CTRL (0x0043)

6.3.29.8.4 Data High of GL_MNG_HWARB_CTRL (0x0044)

6.3.29.9 Reserved (0x0045 - 0x0057)

6.3.29.10 PFPM_APM (0x0058 - 0x006A)

6.3.29.10.1 Starting Address Low at PFPM_APM (0x0058)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFPM_APM, for PF[0]	0xB8080	
3:0	Type	0x2	

6.3.29.10.2 Starting Address High at PFPM_APM (0x0059)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFPM_APM, for PF[0]		

6.3.29.10.3 Attributes at PFPM_APM (0x005A)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.29.10.4 Data Low of PFPM_APM[PF] (0x005B + 2*PF, PF=0...7)

6.3.29.10.5 Data High of PFPM_APM[PF] (0x005C + 2*PF, PF=0...7)

6.3.29.11 PRTGEN_CNF (0x006B - 0x007D)

6.3.29.11.1 Starting Address Low at PRTGEN_CNF (0x006B)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTGEN_CNF, for PRT[0]	0xB8120	
3:0	Type	0x2	

6.3.29.11.2 Starting Address High at PRTGEN_CNF (0x006C)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTGEN_CNF, for PRT[0]		

6.3.29.11.3 Attributes at PRTGEN_CNF (0x006D)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x18	
4:3	Skip	00b	
2:0	Width	000b	

6.3.29.11.4 Data Low of PRTGEN_CNF[PRT] (0x006E + 2*PRT, PRT=0...7)

6.3.29.11.5 Data High of PRTGEN_CNF[PRT] (0x006F + 2*PRT, PRT=0...7)

6.3.29.12 Reserved (0x007E - 0x008D)

6.3.29.13 PRTGEN_CNF2 (0x008E - 0x009D)

6.3.29.13.1 Data Low of PRTGEN_CNF2[PRT] (0x008E + 2*PRT, PRT=0...7)

6.3.29.13.2 Data High of PRTGEN_CNF2[PRT] (0x008F + 2*PRT, PRT=0...7)

6.3.29.14 Reserved (0x009E - 0x00A8)

6.3.29.15 GL_PWR_MODE_CTL (0x00A9 - 0x00AC)

6.3.29.15.1 Address Low at GL_PWR_MODE_CTL (0x00A9)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GL_PWR_MODE_CTL	0xB820C	
3:0	Type	0x1	

6.3.29.15.2 Address High at GL_PWR_MODE_CTL (0x00AA)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GL_PWR_MODE_CTL		

6.3.29.15.3 Data Low of GL_PWR_MODE_CTL (0x00AB)

6.3.29.15.4 Data High of GL_PWR_MODE_CTL (0x00AC)

6.3.30 PSM Preserved Section

Tx-Scheduler parameters.

Table 6-36. PSM Preserved Section Summary Table

Word Offset	Description	Section Reference
0x0000	PSM Preserved Type	6.3.30.1
0x0001	PSM Preserved Length	6.3.30.2
0x0002	Logical Layer Config	6.3.30.3
0x0003 + 1*n, n=0...8	Logical Layer Structure	6.3.30.4
0x000C	Max_RDMA_Qsets	6.3.30.5
0x000D + 4*n, n=0...7	Logical L2/L3 CIR/EIR	6.3.30.6
0x000E + 4*n, n=0...7	Logical L4/L5 CIR/EIR	6.3.30.7
0x000F + 4*n, n=0...7	Logical L6/L7 CIR/EIR	6.3.30.8
0x0010 + 4*n, n=0...7	Logical L8/L9 CIR/EIR	6.3.30.9
0x002D + 1*n, n=0...7	Node Allocation per Layer	6.3.30.10

6.3.30.1 PSM Preserved Type (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x119	Valid values are: 0x119 = PSM preserved

6.3.30.2 PSM Preserved Length (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Block Length		Length in: 2 Bytes unit - 2 First Section -> Word: PSM preserved -> PSM Preserved Type Last Section -> Word: PSM preserved -> Node Allocation per Layer Section length in words.

6.3.30.3 Logical Layer Config (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:14	Reserved	00b	Reserved.
13	L8_L9_SRL	0b	Selects if SRL is used in Physical L8 vs. Physical L9. 0b = L8 1b = L9
12	L6_L7_SRL	0b	Selects if SRL is used in Physical L6 vs. Physical L7. 0b = L6 1b = L7

Bit(s)	Field Name	Default NVM Value	Description
11	L4_L5_SRL	0b	Selects if SRL is used in Physical L4 vs. Physical L5. 0b = L4 1b = L5
10	L2_L3_SRL	0b	Selects if SRL is used in Physical L2 vs. Physical L3. 0b = L2 1b = L3
9	MFP	0b	0b = False 1b = True
8	DCB	1b	0b = False 1b = True
7:4	Number of Logical Layers	0x9	Including port, DCB, MFP and leaf layers.
3:0	Number of Physical Layers	0x9	Device dependent.

6.3.30.4 Logical Layer Structure[n] (0x0003 + 1*n, n=0...8)

Nine structures, one for each logical layer.

Bit(s)	Field Name	Default NVM Value	Description
15:13	Allocation Chunk Size	001b	Valid values are: 000b = N/A 001b = 2 011b = 4 111b = 8
12:0	Maximum Number of Nodes in the Logical Layer	0x8	2 - 4096 depends on the Layer ID and flattening setting based on Logical to Physical layer association.

6.3.30.5 Max_RDMA_Qsets (0x000C)

Bit(s)	Field Name	Default NVM Value	Description
15:10	Reserved	0x0	Reserved.
9:0	Max_RDMA_Qsets_Index	0x1FF	Max index of RDMA QSets.

6.3.30.6 Logical L2/L3 CIR/EIR[n] (0x000D + 4*n, n=0...7)

Allocation to PF of dedicated L2/L3 Rate Limiter profiles.

Bit(s)	Field Name	Default NVM Value	Description
15:8	L3 CIR/EIR	0x1	L3 EIR/CIR RL profiles.
7:0	L2 CIR/EIR	0x1	L2 EIR/CIR RL profiles.

6.3.30.7 Logical L4/L5 CIR/EIR[n] (0x000E + 4*n, n=0...7)

Allocation to PF of dedicated L4/L5 Rate Limiter profiles.

Bit(s)	Field Name	Default NVM Value	Description
15:8	L5 CIR/EIR	0x2	L5 EIR/CIR RL profiles.
7:0	L4 CIR/EIR	0x1	L4 EIR/CIR RL profiles.

6.3.30.8 Logical L6/L7 CIR/EIR[n] (0x000F + 4*n, n=0...7)

Allocation to PF of dedicated L6/L7 Rate Limiter profiles.

Bit(s)	Field Name	Default NVM Value	Description
15:8	L7 CIR/EIR	0x8	L7 EIR/CIR RL profiles. ***Note:*** Max dedicated profiles is 255 out of 256.
7:0	L6 CIR/EIR	0x4	L6 EIR/CIR RL profiles.

6.3.30.9 Logical L8/L9 CIR/EIR[n] (0x0010 + 4*n, n=0...7)

Allocation to PF of dedicated L8/L9 Rate Limiter profiles.

Bit(s)	Field Name	Default NVM Value	Description
15:8	L9 CIR/EIR	0x20	L9 EIR/CIR RL profiles. ***Note:*** Max dedicated profiles is 255 out of 1024.
7:0	L8 CIR/EIR	0x10	L8 EIR/CIR RL profiles. ***Note:*** Max dedicated profiles is 255 out of 512.

6.3.30.10 Node Allocation per Layer[n] (0x002D + 1*n, n=0...7)

Allocation of nodes for each PF. Allocation is uniform to all PFs.

Bit(s)	Field Name	Default NVM Value	Description
15:0	Node Allocation for Layer n	0x1	Number of nodes of Layer n allocated to each valid PF.

6.3.31 MinSrev Section

Table 6-37. MinSrev Section Summary Table

Word Offset	Description	Section Reference
0x0000	MinSrev Module Type	6.3.31.1
0x0001	Length	6.3.31.2
0x0002	Validity	6.3.31.3
0x0003	NVM MinSrev LSB	6.3.31.4
0x0004	NVM MinSrev MSB	6.3.31.5
0x0005	OROM MinSrev LSB	6.3.31.6
0x0006	OROM MinSrev MSB	6.3.31.7

6.3.31.1 MinSrev Module Type (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x130	Valid values are: 0x130 = MinSrev Module

6.3.31.2 Length (0x0001)

Length of section.

Bit(s)	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

6.3.31.3 Validity (0x0002)

Validity of MinSrev words.

Bit(s)	Field Name	Default NVM Value	Description
15:2	Reserved	0x0	Reserved.
1	OROM	0b	OROM MinSrev valid.
0	NVM	0b	NVM MinSrev valid.

6.3.31.4 NVM MinSrev LSB (0x0003)

Bit(s)	Field Name	Default NVM Value	Description
15:0	MinSrev LSB	0x0	NVM MinSrev

6.3.31.5 NVM MinSrev MSB (0x0004)

Bit(s)	Field Name	Default NVM Value	Description
15:0	MinSrev MSB	0x0	NVM MinSrev

6.3.31.6 OROM MinSrev LSB (0x0005)

Bit(s)	Field Name	Default NVM Value	Description
15:0	MinSrev LSB	0x0	OROM MinSrev

6.3.31.7 OROM MinSrev MSB (0x0006)

Bit(s)	Field Name	Default NVM Value	Description
15:0	MinSrev MSB	0x0	OROM MinSrev

6.3.32 PF MAC Address Section

PF/Port MAC Address PFA section.

Table 6-38. PF MAC Address Section Summary Table

Word Offset	Description	Section Reference
0x0000	PCI Serial ID MAC Address Module Type	6.3.32.1
0x0001	Length	6.3.32.2
0x0002	GLPCI_SERL0	6.3.32.3
0x0003	GLPCI_SERL1	6.3.32.4
0x0004	GLPCI_SERH0	6.3.32.5
0x0005	GLPCI_SERH1	6.3.32.6
0x0006	PF MAC Address Module Type	6.3.32.7
0x0007	Section Header	6.3.32.8
0x0008 + 4*n, n=0...7	PFPM_SAL0	6.3.32.9
0x0009 + 4*n, n=0...7	PFPM_SAL1	6.3.32.10
0x000A + 4*n, n=0...7	PFPM_SAH0	6.3.32.11
0x000B + 4*n, n=0...7	PFPM_SAH1	6.3.32.12

6.3.32.1 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

6.3.32.2 Length (0x0001)

Length of MAC Address section.

Bit(s)	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

6.3.32.3 GLPCI_SERL0 (0x0002)

This word is loaded by firmware to GLPCI_SERL 0:15

Bit(s)	Field Name	Default NVM Value	Description
15:0	PFPM_SAL0	0x0	See respective bits in the GLPCI_SERL register.

6.3.32.4 GLPCI_SERL1 (0x00023)

This word is loaded by firmware to GLPCI_SERL 16:31.

Bit(s)	Field Name	Default NVM Value	Description
15:0	PFPM_SAL1	0x0	See respective bits in the GLPCI_SERL register.

6.3.32.5 GLPCI_SERH0 (0x0004)

This word is loaded by firmware to GLPCI_SERH 0:15.

Bit(s)	Field Name	Default NVM Value	Description
15:0	PFPM_SAH0	0x0	See respective bits in the GLPCI_SERH register.

6.3.32.6 GLPCI_SERH1 (0x0005)

This word is loaded by firmware to GLPCI_SERH 16:31.

Bit(s)	Field Name	Default NVM Value	Description
15:0	PFPM_SAH1	0x0	See respective bits in the GLPCI_SERH register.

6.3.32.7 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

6.3.32.8 Section Header (0x0007)

Length of MAC Address section.

Bit(s)	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 -> PFPM_SAH1 Section length in words.

6.3.32.9 PFPM_SAL0[n] (0x0008 + 4*n, n=0...7)

This word is loaded by firmware to the appropriate PRTPM_SAL register, bytes [1,0]. One MAC Address per enabled PF/port. This is a per-device, per-enabled-PF value allocated by manufacturing.

Bit(s)	Field Name	Default NVM Value	Description
15:0	PFPM_SAL0	0x0	See respective bits in the PRTPM_SAL register.

6.3.32.10 PFPM_SAL1[n] (0x0009 + 4*n, n=0...7)

This register is loaded by firmware to the appropriate PRTPM_SAL register, bytes [3:2]. One MAC Address per enabled PF. This is a per-device, per-enabled-PF value allocated by manufacturing.

Bit(s)	Field Name	Default NVM Value	Description
15:0	PFPM_SAL1	0x0	See respective bits in the PRTPM_SAL register.

6.3.32.11 PFPM_SAH0[n] (0x000A + 4*n, n=0...7)

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [1,0]. One MAC Address per enabled PF/port. This is a per-device, per-enabled-PF value allocated by manufacturing.

Bit(s)	Field Name	Default NVM Value	Description
15:0	PFPM_SAH0	0x0	See respective bits in the PRTPM_SAH register.

6.3.32.12 PFPM_SAH1[n] (0x000B + 4*n, n=0...7)

This register is loaded by firmware to the appropriate PRTPM_SAH register, bytes [3:2]. One MAC Address per enabled PF/port. This is a per-device, per-enabled-PF value allocated by manufacturing.

Bit(s)	Field Name	Default NVM Value	Description
15:0	PFPM_SAH1	0x0	See respective bits in the PRTPM_SAH register.

6.3.33 MNG MAC Address Section

Manageability MAC Address PFA section.

Table 6-39. MNG MAC Address Section Summary Table

Word Offset	Description	Section Reference
0x0000	MNG MAC Address Module Type	6.3.33.1
0x0001	Section Header - Length	6.3.33.2
0x0002 + 3*n, n=0...31	LAN Ethernet MAC Address (LSB) MMAL	6.3.33.3
0x0003 + 3*n, n=0...31	LAN Ethernet MAC Address (Mid) MMAL	6.3.33.4
0x0004 + 3*n, n=0...31	LAN Ethernet MAC Address (MSB) MMAH	6.3.33.5

6.3.33.1 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

6.3.33.2 Section Header - Length (0x0001)

Bit(s)	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.

6.3.33.3 LAN Ethernet MAC Address (LSB) MMAL[n] (0x0002 + 3*n, n=0...31)

This word is loaded by the firmware 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.

Bit(s)	Field Name	Default NVM Value	Description
15:0	Ethernet MAC Address, Byte 1	0xFF	
7:0	Ethernet MAC Address, Byte 0	0xFF	

6.3.33.4 LAN Ethernet MAC Address (Mid) MMAL[n] (0x0003 + 3*n, n=0...31)

This word is loaded by the firmware 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.

Bit(s)	Field Name	Default NVM Value	Description
15:0	Ethernet MAC Address, Byte 3	0xFF	
7:0	Ethernet MAC Address, Byte 2	0xFF	

6.3.33.5 LAN Ethernet MAC Address (MSB) MMAH[n] (0x0004 + 3*n, n=0...31)

This word is loaded by the firmware to the MMAH 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.

Bit(s)	Field Name	Default NVM Value	Description
15:0	Ethernet MAC Address, Byte 5	0xFF	
7:0	Ethernet MAC Address, Byte 4	0xFF	

6.3.34 FW Logging Defaults Section

UART debug defaults.

Table 6-40. FW Logging Defaults Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type	6.3.34.1
0x0001	Section Length	6.3.34.2
0x0002	UART Control	6.3.34.3
0x0003	Module Logging Enable [n=0]	6.3.34.4
0x0004	Module Logging Enable [n=1]	6.3.34.5
0x0005	Module Logging Enable [n=2]	6.3.34.6
0x0006	Module Logging Enable [n=3]	6.3.34.7
0x0007	Module Logging Enable [n=4]	6.3.34.8
0x0008	Module Logging Enable [n=5]	6.3.34.9
0x0009	Module Logging Enable [n=6]	6.3.34.10
0x000A	Module Logging Enable [n=7]	6.3.34.11

6.3.34.1 Sub Module Type (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x11D	Valid values are: 0x11D = UART Debug

6.3.34.2 Section Length (0x0001)

Bit(s)	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.

6.3.34.3 UART Control (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:1	Reserved	0x0	Reserved.
0	UART Logging Enable	0b	0b = Disabled 1b = Enabled

6.3.34.4 Module Logging Enable [n=0] (0x0003)

Bit(s)	Field Name	Default NVM Value	Description
15	Module 4n+3 Error Logging Enable	0b	0b = Disabled 1b = Enabled
14	Module 4n+3 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
13	Module 4n+3 Init Logging Enable	0b	0b = Disabled 1b = Enabled
12	Module 4n+3 Info Logging Enable	0b	0b = Disabled 1b = Enabled
11	Module 4n+2 Error Logging Enable	0b	0b = Disabled 1b = Enabled
10	Module 4n+2 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
9	Module 4n+2 Init Logging Enable	0b	0b = Disabled 1b = Enabled
8	Module 4n+2 Info Logging Enable	0b	0b = Disabled 1b = Enabled
7	Module 4n+1 Error Logging Enable	0b	0b = Disabled 1b = Enabled
6	Module 4n+1 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
5	Module 4n+1 Init Logging Enable	0b	0b = Disabled 1b = Enabled
4	Module 4n+1 Info Logging Enable	0b	0b = Disabled 1b = Enabled
3	Module 4n+0 Error Logging Enable	0b	0b = Disabled 1b = Enabled
2	Module 4n+0 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
1	Module 4n+0 Init Logging Enable	0b	0b = Disabled 1b = Enabled
0	Module 4n+0 Info Logging Enable	0b	0b = Disabled 1b = Enabled

6.3.34.5 Module Logging Enable [n=1] (0x0004)

Bit(s)	Field Name	Default NVM Value	Description
15	Module 4n+3 Error Logging Enable	0b	0b = Disabled 1b = Enabled
14	Module 4n+3 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
13	Module 4n+3 Init Logging Enable	0b	0b = Disabled 1b = Enabled
12	Module 4n+3 Info Logging Enable	0b	0b = Disabled 1b = Enabled

Bit(s)	Field Name	Default NVM Value	Description
11	Module 4n+2 Error Logging Enable	0b	0b = Disabled 1b = Enabled
10	Module 4n+2 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
9	Module 4n+2 Init Logging Enable	0b	0b = Disabled 1b = Enabled
8	Module 4n+2 Info Logging Enable	0b	0b = Disabled 1b = Enabled
7	Module 4n+1 Error Logging Enable	0b	0b = Disabled 1b = Enabled
6	Module 4n+1 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
5	Module 4n+1 Init Logging Enable	0b	0b = Disabled 1b = Enabled
4	Module 4n+1 Info Logging Enable	0b	0b = Disabled 1b = Enabled
3	Module 4n+0 Error Logging Enable	0b	0b = Disabled 1b = Enabled
2	Module 4n+0 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
1	Module 4n+0 Init Logging Enable	0b	0b = Disabled 1b = Enabled
0	Module 4n+0 Info Logging Enable	0b	0b = Disabled 1b = Enabled

6.3.34.6 Module Logging Enable [n=2] (0x0005)

Bit(s)	Field Name	Default NVM Value	Description
15	Module 4n+3 Error Logging Enable	0b	0b = Disabled 1b = Enabled
14	Module 4n+3 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
13	Module 4n+3 Init Logging Enable	0b	0b = Disabled 1b = Enabled
12	Module 4n+3 Info Logging Enable	0b	0b = Disabled 1b = Enabled
11	Module 4n+2 Error Logging Enable	0b	0b = Disabled 1b = Enabled
10	Module 4n+2 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
9	Module 4n+2 Init Logging Enable	0b	0b = Disabled 1b = Enabled
8	Module 4n+2 Info Logging Enable	0b	0b = Disabled 1b = Enabled
7	Module 4n+1 Error Logging Enable	0b	0b = Disabled 1b = Enabled
6	Module 4n+1 Flow Logging Enable	0b	0b = Disabled 1b = Enabled

Bit(s)	Field Name	Default NVM Value	Description
5	Module 4n+1 Init Logging Enable	0b	0b = Disabled 1b = Enabled
4	Module 4n+1 Info Logging Enable	0b	0b = Disabled 1b = Enabled
3	Module 4n+0 Error Logging Enable	0b	0b = Disabled 1b = Enabled
2	Module 4n+0 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
1	Module 4n+0 Init Logging Enable	0b	0b = Disabled 1b = Enabled
0	Module 4n+0 Info Logging Enable	0b	0b = Disabled 1b = Enabled

6.3.34.7 Module Logging Enable [n=3] (0x0006)

Bit(s)	Field Name	Default NVM Value	Description
15	Module 4n+3 Error Logging Enable	0b	0b = Disabled 1b = Enabled
14	Module 4n+3 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
13	Module 4n+3 Init Logging Enable	0b	0b = Disabled 1b = Enabled
12	Module 4n+3 Info Logging Enable	0b	0b = Disabled 1b = Enabled
11	Module 4n+2 Error Logging Enable	0b	0b = Disabled 1b = Enabled
10	Module 4n+2 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
9	Module 4n+2 Init Logging Enable	0b	0b = Disabled 1b = Enabled
8	Module 4n+2 Info Logging Enable	0b	0b = Disabled 1b = Enabled
7	Module 4n+1 Error Logging Enable	0b	0b = Disabled 1b = Enabled
6	Module 4n+1 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
5	Module 4n+1 Init Logging Enable	0b	0b = Disabled 1b = Enabled
4	Module 4n+1 Info Logging Enable	0b	0b = Disabled 1b = Enabled
3	Module 4n+0 Error Logging Enable	0b	0b = Disabled 1b = Enabled
2	Module 4n+0 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
1	Module 4n+0 Init Logging Enable	0b	0b = Disabled 1b = Enabled
0	Module 4n+0 Info Logging Enable	0b	0b = Disabled 1b = Enabled

6.3.34.8 Module Logging Enable [n=4] (0x0007)

Bit(s)	Field Name	Default NVM Value	Description
15	Module 4n+3 Error Logging Enable	0b	0b = Disabled 1b = Enabled
14	Module 4n+3 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
13	Module 4n+3 Init Logging Enable	0b	0b = Disabled 1b = Enabled
12	Module 4n+3 Info Logging Enable	0b	0b = Disabled 1b = Enabled
11	Module 4n+2 Error Logging Enable	0b	0b = Disabled 1b = Enabled
10	Module 4n+2 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
9	Module 4n+2 Init Logging Enable	0b	0b = Disabled 1b = Enabled
8	Module 4n+2 Info Logging Enable	0b	0b = Disabled 1b = Enabled
7	Module 4n+1 Error Logging Enable	0b	0b = Disabled 1b = Enabled
6	Module 4n+1 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
5	Module 4n+1 Init Logging Enable	0b	0b = Disabled 1b = Enabled
4	Module 4n+1 Info Logging Enable	0b	0b = Disabled 1b = Enabled
3	Module 4n+0 Error Logging Enable	0b	0b = Disabled 1b = Enabled
2	Module 4n+0 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
1	Module 4n+0 Init Logging Enable	0b	0b = Disabled 1b = Enabled
0	Module 4n+0 Info Logging Enable	0b	0b = Disabled 1b = Enabled

6.3.34.9 Module Logging Enable [n=5] (0x0008)

Bit(s)	Field Name	Default NVM Value	Description
15	Module 4n+3 Error Logging Enable	0b	0b = Disabled 1b = Enabled
14	Module 4n+3 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
13	Module 4n+3 Init Logging Enable	0b	0b = Disabled 1b = Enabled
12	Module 4n+3 Info Logging Enable	0b	0b = Disabled 1b = Enabled

Bit(s)	Field Name	Default NVM Value	Description
11	Module 4n+2 Error Logging Enable	0b	0b = Disabled 1b = Enabled
10	Module 4n+2 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
9	Module 4n+2 Init Logging Enable	0b	0b = Disabled 1b = Enabled
8	Module 4n+2 Info Logging Enable	0b	0b = Disabled 1b = Enabled
7	Module 4n+1 Error Logging Enable	0b	0b = Disabled 1b = Enabled
6	Module 4n+1 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
5	Module 4n+1 Init Logging Enable	0b	0b = Disabled 1b = Enabled
4	Module 4n+1 Info Logging Enable	0b	0b = Disabled 1b = Enabled
3	Module 4n+0 Error Logging Enable	0b	0b = Disabled 1b = Enabled
2	Module 4n+0 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
1	Module 4n+0 Init Logging Enable	0b	0b = Disabled 1b = Enabled
0	Module 4n+0 Info Logging Enable	0b	0b = Disabled 1b = Enabled

6.3.34.10 Module Logging Enable [n=6] (0x0009)

Bit(s)	Field Name	Default NVM Value	Description
15	Module 4n+3 Error Logging Enable	0b	0b = Disabled 1b = Enabled
14	Module 4n+3 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
13	Module 4n+3 Init Logging Enable	0b	0b = Disabled 1b = Enabled
12	Module 4n+3 Info Logging Enable	0b	0b = Disabled 1b = Enabled
11	Module 4n+2 Error Logging Enable	0b	0b = Disabled 1b = Enabled
10	Module 4n+2 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
9	Module 4n+2 Init Logging Enable	0b	0b = Disabled 1b = Enabled
8	Module 4n+2 Info Logging Enable	0b	0b = Disabled 1b = Enabled
7	Module 4n+1 Error Logging Enable	0b	0b = Disabled 1b = Enabled
6	Module 4n+1 Flow Logging Enable	0b	0b = Disabled 1b = Enabled

Bit(s)	Field Name	Default NVM Value	Description
5	Module 4n+1 Init Logging Enable	0b	0b = Disabled 1b = Enabled
4	Module 4n+1 Info Logging Enable	0b	0b = Disabled 1b = Enabled
3	Module 4n+0 Error Logging Enable	0b	0b = Disabled 1b = Enabled
2	Module 4n+0 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
1	Module 4n+0 Init Logging Enable	0b	0b = Disabled 1b = Enabled
0	Module 4n+0 Info Logging Enable	0b	0b = Disabled 1b = Enabled

6.3.34.11 Module Logging Enable [n=7] (0x000A)

Bit(s)	Field Name	Default NVM Value	Description
15	Module 4n+3 Error Logging Enable	0b	0b = Disabled 1b = Enabled
14	Module 4n+3 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
13	Module 4n+3 Init Logging Enable	0b	0b = Disabled 1b = Enabled
12	Module 4n+3 Info Logging Enable	0b	0b = Disabled 1b = Enabled
11	Module 4n+2 Error Logging Enable	0b	0b = Disabled 1b = Enabled
10	Module 4n+2 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
9	Module 4n+2 Init Logging Enable	0b	0b = Disabled 1b = Enabled
8	Module 4n+2 Info Logging Enable	0b	0b = Disabled 1b = Enabled
7	Module 4n+1 Error Logging Enable	0b	0b = Disabled 1b = Enabled
6	Module 4n+1 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
5	Module 4n+1 Init Logging Enable	0b	0b = Disabled 1b = Enabled
4	Module 4n+1 Info Logging Enable	0b	0b = Disabled 1b = Enabled
3	Module 4n+0 Error Logging Enable	0b	0b = Disabled 1b = Enabled
2	Module 4n+0 Flow Logging Enable	0b	0b = Disabled 1b = Enabled
1	Module 4n+0 Init Logging Enable	0b	0b = Disabled 1b = Enabled
0	Module 4n+0 Info Logging Enable	0b	0b = Disabled 1b = Enabled

6.3.35 1588 Parameters Section

1588 functionality parameters.

Table 6-41. 1588 Parameters Section Summary Table

Word Offset	Description	Section Reference
0x0000	Type	6.3.35.1
0x0001	Length	6.3.35.2
0x0002	1588 Timer Ownership	6.3.35.3
0x0003	1588 Functionality Enablement 0-7	6.3.35.4
0x0004	1588 Functionality Enablement 8-15	6.3.35.5
0x0005	1588 Functionality Enablement 16-19	6.3.35.6

6.3.35.1 Type (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x4F	Valid values are: 0x4F = 1588 Parameters

6.3.35.2 Length (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Block Length		Length in: 2 Bytes unit - 2 First Section -> Word: 1588 parameters -> Type Last Section -> Word: 1588 parameters -> 1588 Functionality Enablement 16-19 Section length in words.

6.3.35.3 1588 Timer Ownership (0x0002)

1588 timer ownership by PF.

Bit(s)	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	Reserved.
11	T1owned	0b	0b = Disabled 1b = Enabled
10:8	T1owner	000b	Timer 1 PF owner. 000b = Owned by PF 0 001b = Owned by PF 1 010b = Owned by PF 2 011b = Owned by PF 3 100b = Owned by PF 4 101b = Owned by PF 5 011b = Owned by PF 6 111b = Owned by PF 7

Bit(s)	Field Name	Default NVM Value	Description
7:4	Reserved	0x0	Reserved.
3	T0owned	0b	0b = Disabled 1b = Enabled
2:0	T0owner	000b	Timer 0 PF owner. 000b = Owned by PF 0 001b = Owned by PF 1 010b = Owned by PF 2 011b = Owned by PF 3 100b = Owned by PF 4 101b = Owned by PF 5 011b = Owned by PF 6 111b = Owned by PF 7

6.3.35.4 1588 Functionality Enablement 0-7 (0x0003)

1588 functionality enablement per port.

Bit(s)	Field Name	Default NVM Value	Description
15:14	P7	00b	Port 7 same list as Port 0.
13:12	P6	00b	Port 6 same list as Port 0.
11:10	P5	00b	Port 5 same list as Port 0.
9:8	P4	00b	Port 4 same list as Port 0.
7:6	P3	00b	Port 3 same list as Port 0.
5:4	P2	00b	Port 2 same list as Port 0.
3:2	P1	00b	Port 1 same list as Port 0.
1:0	P0	00b	Port 0. 00b = Disabled 01b = Reserved 10b = Timer 0 11b = Timer 1

6.3.35.5 1588 Functionality Enablement 8-15 (0x0004)

1588 functionality enablement per port.

Bit(s)	Field Name	Default NVM Value	Description
15:14	P15	00b	Port 15 same list as Port 8.
13:12	P14	00b	Port 14 same list as Port 8.
11:10	P13	00b	Port 13 same list as Port 8.
9:8	P12	00b	Port 12 same list as Port 8.
7:6	P11	00b	Port 11 same list as Port 8.
5:4	P10	00b	Port 10 same list as Port 8.
3:2	P9	00b	Port 9 same list as Port 8.

Bit(s)	Field Name	Default NVM Value	Description
1:0	P8	00b	Port 8. 00b = Disabled 01b = Reserved 10b = Timer 0 11b = Timer 1

6.3.35.6 1588 Functionality Enablement 16-19 (0x0005)

1588 functionality enablement per port.

Bit(s)	Field Name	Default NVM Value	Description
15:8	Reserved	0x0	Reserved.
7:6	P19	00b	Port 19 same list as Port 16.
5:4	P18	00b	Port 18 same list as Port 16.
3:2	P17	00b	Port 17 same list as Port 16.
1:0	P16	00b	Port 16. 00b = Disabled 01b = Reserved 10b = Timer 0 11b = Timer 1

6.3.36 MD Link Topology Section

This is the Ethernet Link topology description of the motherboard. Link topology is described in Section 3.3.

Table 6-42. MD Link Topology Section Summary Table

Word Offset	Description	Section Reference
0x0000	FW MNG Link Topology Module Type	6.3.36.1
0x0001	FW MNG Link Topology Module Length	6.3.36.2
0x0002	Generic Info	6.3.36.3
0x0003	Netlist Version	6.3.36.4
0x0004 + 1*n, n=0...15	Pair PHY Type	6.3.36.5
0x0014	Port Bitmap 0	6.3.36.6
0x0015	Port Bitmap 1	6.3.36.7

6.3.36.1 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

6.3.36.2 FW MNG Link Topology Module Length (0x0001)

Bit(s)	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.

6.3.36.3 Generic Info (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:8	Netlist Map Version	0x1	The version of the Netlist field definition.
7:1	Reserved	0x0	Reserved.

Bit(s)	Field Name	Default NVM Value	Description
0	Load Mode	1b	Netlist Loading Mode 0b = Normal Mode 1b = Resolution Mode Controls the steps taken at netlist load time. Valid only for the motherboard's netlist. 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.

6.3.36.4 Netlist Version (0x0003)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Netlist Version	0x0	The version of the netlist.

6.3.36.5 Pair PHY Type[n] (0x0004 + 1*n, n=0...15)

Bit(s)	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.

6.3.36.6 Port Bitmap 0 (0x0014)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Port Bitmap 0	0x0	

6.3.36.7 Port Bitmap 1 (0x0015)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Reserved	0x0	Reserved.
3:0	Port Bitmap 1	0x0	

6.3.37 LLDP Preserved Section

Table 6-43. LLDP Preserved Section Summary Table

Word Offset	Description	Section Reference
0x0000	Type	6.3.37.1
0x0001	Length	6.3.37.2
0x0002	LLDP Admin Status 0	6.3.37.3
0x0003	LLDP Admin Status 1	6.3.37.4

6.3.37.1 Type (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x129	Valid values are: 0x129 = LLDP Preserved

6.3.37.2 Length (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Block Length		Length in: 2 Bytes unit - 2 First Section -> Word: LLDP Preserved -> Type Last Section -> Word: LLDP Preserved -> LLDP Admin Status 1 Section length in words.

6.3.37.3 LLDP Admin Status 0 (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.

Bit(s)	Field Name	Default NVM Value	Description
15:12	Port 3	0xF	Defines status of LLDP agent. Applies to LAN Port 3.
11:8	Port 2	0xF	Defines status of LLDP agent. Applies to LAN Port 2.
7:4	Port 1	0xF	Defines status of LLDP agent. Applies to LAN Port 1.
3:0	Port 0	0xF	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.37.4 LLDP Admin Status 1 (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.

Bit(s)	Field Name	Default NVM Value	Description
15:12	Port 7	0xF	Defines status of LLDP agent. Applies to LAN Port 7.
11:8	Port 6	0xF	Defines status of LLDP agent. Applies to LAN Port 6.
7:4	Port 5	0xF	Defines status of LLDP agent. Applies to LAN Port 5.
3:0	Port 4	0xF	Defines status of LLDP agent. Applies to LAN Port 4.

6.3.38 RDE Module Section

LLDP Preserved.

Table 6-44. RDE Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	Type	6.3.38.1
0x0001	Length	6.3.38.2
0x0002 + 1*n, n=0...31	AssetTag	6.3.38.3

6.3.38.1 Type (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x131	Valid values are: 0x131 = RDE Module

6.3.38.2 Length (0x0001)

Bit(s)	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 -> AssetTag Section length in words.

6.3.38.3 AssetTag[n] (0x0002 + 1*n, n=0...31)

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.

Bit(s)	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.39 Identical Content as PLDM Header ComponentImageSetVersionString Section

Table 6-45. Identical Content as PLDM Header ComponentImageSetVersionString Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type	6.3.39.1
0x0001	Length	6.3.39.2
0x0002	ComponentImageSetVersionString- FW Component Prefix	6.3.39.3
0x0003	ComponentImageSetVersionString - FW Component Version 0	6.3.39.4
0x0004	ComponentImageSetVersionString - FW Component Version 1	6.3.39.5
0x0005	ComponentImageSetVersionString - FW Component Version 2	6.3.39.6
0x0006	ComponentImageSetVersionString - FW Component Version 3	6.3.39.7
0x0007	ComponentImageSetVersionString - OROM Component Prefix	6.3.39.8
0x0008	ComponentImageSetVersionString - OROM Component Version 0	6.3.39.9
0x0009	ComponentImageSetVersionString - OROM Component Version 1	6.3.39.10
0x000A	ComponentImageSetVersionString - OROM Component Version 2	6.3.39.11
0x000B	ComponentImageSetVersionString - OROM Component Version 3	6.3.39.12
0x000C	ComponentImageSetVersionString - Netlist Component Prefix	6.3.39.13
0x000D	ComponentImageSetVersionString - Netlist Component Version 0	6.3.39.14
0x000E	ComponentImageSetVersionString - Netlist Component Version 1	6.3.39.15
0x000F	ComponentImageSetVersionString - Netlist Component Version 2	6.3.39.16
0x0010	ComponentImageSetVersionString - Netlist Component Version 3	6.3.39.17
0x0011	ComponentImageSetVersionString - Null Padding	6.3.39.18

6.3.39.1 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

6.3.39.2 Length (0x0001)

Bit(s)	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

6.3.39.3 ComponentImageSetVersionString- FW Component Prefix (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - FW Component Prefix	0x3A4E	"N:" in ASCII.

6.3.39.4 ComponentImageSetVersionString - FW Component Version 0 (0x0003)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - FW Component Version 0	0x3030	Firmware component version 0. ASCII presentation of firmware comparison stamp. Values are zero (in ASCII) padded.

6.3.39.5 ComponentImageSetVersionString - FW Component Version 1 (0x0004)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - FW Component Version 1	0x3030	Firmware component version 1. ASCII presentation of firmware comparison stamp. Values are zero (in ASCII) padded.

6.3.39.6 ComponentImageSetVersionString - FW Component Version 2 (0x0005)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - FW Component Version 2	0x3030	Firmware component version 2. ASCII presentation of firmware comparison stamp. Values are zero (in ASCII) padded.

6.3.39.7 ComponentImageSetVersionString - FW Component Version 3 (0x0006)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - FW Component Version 3	0x3030	Firmware component version 3. ASCII presentation of firmware comparison stamp. Values are zero (in ASCII) padded.

6.3.39.8 ComponentImageSetVersionString - OROM Component Prefix (0x0007)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - OROM Component Prefix	0x3A4F	"O:" in ASCII.

6.3.39.9 ComponentImageSetVersionString - OROM Component Version 0 (0x0008)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - OROM Component Version 0	0x3030	OROM component version 0. ASCII presentation of OROM comparison stamp. Values are zero (in ASCII) padded.

6.3.39.10 ComponentImageSetVersionString - OROM Component Version 1 (0x0009)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - OROM Component Version 1	0x3030	OROM component version 1. ASCII presentation of OROM comparison stamp. Values are zero (in ASCII) padded.

6.3.39.11 ComponentImageSetVersionString - OROM Component Version 2 (0x000A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - OROM Component Version 2	0x3030	OROM component version 2. ASCII presentation of OROM comparison stamp. Values are zero (in ASCII) padded.

6.3.39.12 ComponentImageSetVersionString - OROM Component Version 3 (0x000B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - OROM Component Version 3	0x3030	OROM component version 3. ASCII presentation of OROM comparison stamp. Values are zero (in ASCII) padded.

6.3.39.13 ComponentImageSetVersionString - Netlist Component Prefix (0x000C)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSet VersionString - Netlist Component Prefix	0x3A54	"T:" in ASCII.

6.3.39.14 ComponentImageSetVersionString - Netlist Component Version 0 (0x000D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - Netlist component version 0	0x3030	Netlist component version 0. ASCII presentation of Netlist comparison stamp. Values are zero (in ASCII) padded.

6.3.39.15 ComponentImageSetVersionString - Netlist Component Version 1 (0x000E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - Netlist Component Version 1	0x3030	Netlist component version 1. ASCII presentation of Netlist comparison stamp. Values are zero (in ASCII) padded.

6.3.39.16 ComponentImageSetVersionString - Netlist Component Version 2 (0x000F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - Netlist Component Version 2	0x3030	Netlist component version 2. ASCII presentation of Netlist comparison stamp. Values are zero (in ASCII) padded.

6.3.39.17 ComponentImageSetVersionString - Netlist Component Version 3 (0x0010)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - Netlist Component Version 3	0x3030	Netlist component version 3. ASCII presentation of Netlist comparison stamp. Values are zero (in ASCII) padded.

6.3.39.18 ComponentImageSetVersionString - Null Padding (0x0011)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - Null Padding	0x0	

6.3.40 Software Checksum Module Section

The Software 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 software 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.

Table 6-46. Software Checksum Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type - Checksum	6.3.40.1
0x0001	Checksum Module Length	6.3.40.2
0x0002	Checksum	6.3.40.3

6.3.40.1 Sub Module Type - Checksum (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x3F	Valid values are: 0x3F = Checksum

6.3.40.2 Checksum Module Length (0x0001)

Bit(s)	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

6.3.40.3 Checksum (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Checksum		Checksum word. Calculated so that checksum over PFA is 0xBABA.

6.3.41 RDMA Control Section

Table 6-47. RDMA Control Section Summary Table

Word Offset	Description	Section Reference
0x0000	RDMA Control Module Type	6.3.41.1
0x0001	Length	6.3.41.2
0x0002	RDMA Control Settings	6.3.41.3

6.3.41.1 RDMA Control Module Type (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x135	Valid values are: 0x135 = RDMA Control Module

6.3.41.2 Length (0x0001)

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 2 First Section -> Word: RDMA Control -> RDMA Control Module Type Last Section -> Word: RDMA Control -> RDMA Control Settings

6.3.41.3 RDMA Control Settings (0x0002)

Bits	Field Name	Default NVM Value	Description
15:5	Reserved	0x0	Reserved.
4	High Port Count RDMA Control Override Value	0b	Enable or disable. 0b = Disabled 1b = Enabled
3	High Port Count RDMA Control Override Enable	0b	Enable/Disable alternative configuration request for 5 ports or higher. 0b = Disabled 1b = Enabled
2	Low Port Count RDMA Control Override Value	0b	Enable or disable. 0b = Disabled 1b = Enabled
1	Low port count RDMA Control Override Enable	0b	Enable/Disable alternative configuration request for 4 ports or lower. 0b = Disabled 1b = Enabled
0	RDMA Topology Setting	0b	Enable or disable. 0b = Disabled 1b = Enabled

6.3.42 Link Default Override Mask Section

Table 6-48. Link Default Override Mask Section Summary Table

Word Offset	Description	Section Reference
0x0000	Link Default Override Mask Type	6.3.42.1
0x0001	Length	6.3.42.2
0x0002 + 10*n, n=0...7	Port Options 0	6.3.42.3
0x0003 + 10*n, n=0...7	Port Options 1	6.3.42.4
0x0004 + 10*n, n=0...7	Port PHY Types 0	6.3.42.5
0x0005 + 10*n, n=0...7	Port PHY Types 1	6.3.42.6
0x0006 + 10*n, n=0...7	Port PHY Types 2	6.3.42.7
0x0007 + 10*n, n=0...7	Port PHY Types 3	6.3.42.8
0x0008 + 10*n, n=0...7	Port PHY Types 4	6.3.42.9
0x0009 + 10*n, n=0...7	Port PHY Types 5	6.3.42.10
0x000A + 10*n, n=0...7	Port PHY Types 6	6.3.42.11
0x000B + 10*n, n=0...7	Port PHY Types 7	6.3.42.12

6.3.42.1 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

6.3.42.2 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

6.3.42.3 Port Options 0[n] (0x0002 + 10*n, n=0...7)

Port link options.

Bits	Field Name	Default NVM Value	Description
15	Auto FEC Enable	1b	When *Override Enable* bit is set, the *Auto FEC Enable* bit is taken from here and not from the topology netlist.
14	LESM Enable	1b	When *Override Enable* bit is set, the *LESM Enable* bit is taken from here and not from the topology netlist.
13:10	Reserved	0xF	Reserved.
9:8	PAUSE Ability	00b	When *Override Enable* bit is set, the *PAUSE Ability* bits are taken from here and not from the topology netlist.
7:6	Reserved	11b	Reserved.
5	EEE Enable Override	0b	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. 0b = Disable 1b = Enable
4	Disable Automatic Link on Startup Override	0b	When *Override Enable* bit is set, the "Disable Automatic Link on Startup" feature is taken from here and not from the link topology. 0b = Automatic Link on Startup feature is enabled. 1b = Automatic Link on Startup feature is disabled.
3	Override Enable	0b	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. 0b = Disable 1b = Enable
2	Port Disable Behavior Mode	0b	Controls how to disable the port when requested. 0b = Disable only the host. 1b = Disables all usages.
1	EPCT ability to Change Lenient/Strict Enable	0b	EPCT ability to change Lenient/Strict mode.
0	Lenient/Strict Mode	0b	0b - Lenient 1b = Strict

6.3.42.4 Port Options 1[n] (0x0003 + 10*n, n=0...7)

Port PHY options.

Bits	Field Name	Default NVM Value	Description
15:8	Reserved	0xFF	Reserved.
7	FIRE_CODE_25_ABILITY	1b	Controls KR FEC ability advertisement for 25G KR1/CR1. 0b = EC Disabled 1b = FEC Enabled (advertised)
6	RS_528_ABILITY	1b	Controls RS FEC ability advertisement for 25G KR1/CR1. 0b = FEC Disabled 1b = FEC Enabled (advertised)
5	Reserved	0b	Reserved.
4	RS_544_REQUEST	0b	Controls RS FEC 544 capability request for 25G/50Glanes KR/KR-S/KR1/CR/CR-S/CR1.

Bits	Field Name	Default NVM Value	Description
3	FIRE_CODE_25_REQUEST	1b	Controls KR FEC capability request for 25G lanes KR/KR-S/KR1/CR/CR-S/CR1. 0b = FEC Disabled 1b = FEC Enabled (requested)
2	RS_528_REQUEST	1b	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	0b	Controls FEC capability request for 10G KR and 40G KR4/CR4. 0b = FEC Disabled 1b = FEC Enabled (requested)
0	FIRE_CODE_10_ABILITY Enable	0b	Controls FEC capability advertisement for 10G KR and 40G KR4/CR4. 0b = FEC Disabled 1b = FEC Enabled (advertised)

6.3.42.5 Port PHY Types 0[n] (0x0004 + 10*n, n=0...7)

PHY types. 8 Words per port. 8 ports.

Bits	Field Name	Default NVM Value	Description
15	10GBASE-LR	1b	
14	10GBASE-SR	1b	
13	10G-SFI-DA	1b	
12	10GBASE-T	1b	
11	5GBASE-KR	1b	
10	5GBASE-T	1b	
9	2.5GBASE-KX	1b	
8	2.5GBASE-X	1b	
7	2.5GBASE-T	1b	
6	1G-SGMII	1b	
5	1000BASE-KX	1b	
4	1000BASE-LX	1b	
3	1000BASE-SX	1b	
2	1000BASE-T	1b	
1	100M-SGMII	1b	
0	100BASE-TX	1b	

6.3.42.6 Port PHY Types 1[n] (0x0005 + 10*n, n=0...7)

Bits	Field Name	Default NVM Value	Description
15	40GBASE-SR4	1b	
14	40GBASE-CR4	1b	
13	25G-AUI-C2C	1b	
12	25G-AUI-AOC\ACC	1b	
11	25GBASE-KR1	1b	
10	25GBASE-KR-S	1b	
9	25GBASE-KR	1b	
8	25GBASE-LR	1b	
7	25GBASE-SR	1b	
6	25GBASE-CR1	1b	
5	25GBASE-CR-S	1b	
4	25GBASE-CR	1b	
3	25GBASE-T	1b	
2	10G-SFI-C2C	1b	
1	10G-SFI-AOC\ACC	1b	
0	10GBASE-KR\CR1	1b	

6.3.42.7 Port PHY Types 2[n] (0x0006 + 10*n, n=0...7)

Bits	Field Name	Default NVM Value	Description
15	50GBASE-LR	1b	
14	50GBASE-FR	1b	
13	50GBASE-SR	1b	
12	50GBASE-CP	1b	
11	50G-AUI2	1b	
10	50G-AUI2-AOC\ACC	1b	
9	50G-LAUI2	1b	
8	50G-LAUI2-AOC\ACC	1b	
7	50GBASE-KR2	1b	
6:5	Reserved	11b	Reserved.
4	50GBASE-CR2	1b	
3	40G-XLAUI	1b	
2	40G-XLAUI-AOC\ACC	1b	
1	40GBASE-KR4	1b	
0	40GBASE-LR4	1b	

6.3.42.8 Port PHY Types 3[n] (0x0007 + 10*n, n=0...7)

Bits	Field Name	Default NVM Value	Description
15	100GBASE-DR	1b	
14	100GBASE-SR2	1b	
13	100GBASE-CP2	1b	
12	100GBASE-KP-PAM4	1b	
11	Reserved	1b	Reserved.
10	100G-AUI4	1b	
9	100G-AUI4-AOC\ACC	1b	
8	100G-CAUI4	1b	
7	100G-CAUI-AOC\ACC	1b	
6	100GBASE-KR4	1b	
5	100GBASE-LR4	1b	
4	100GBASE-SR4	1b	
3	100GBASE-CR4	1b	
2	50G-AUI1	1b	
1	50G-AUI1-AOC\ACC	1b	
0	50GBASE-KR-PAM4	1b	

6.3.42.9 Port PHY Types 4[n] (0x0008 + 10*n, n=0...7)

Bits	Field Name	Default NVM Value	Description
15	400GBASE-FR8	1b	
14	200G-AUI8	1b	
13	200G-AUI8-AOC\ACC	1b	
12	200G-AUI4	1b	
11	200G-AUI4-AOC\ACC	1b	
10	200GBASE-KR4-PAM4	1b	
9	200GBASE-DR4	1b	
8	200GBASE-LR4	1b	
7	200GBASE-FR4	1b	
6	200GBASE-SR4	1b	
5	200GBASE-CR4-PAM4	1b	
4	100G-AUI2	1b	
3	100G-AUI2-AOC\ACC	1b	
2:1	Reserved	11b	Reserved.
0	100GBASE-KR2-PAM4	1b	

6.3.42.10 Port PHY Types 5[n] (0x0009 + 10*n, n=0...7)

Bits	Field Name	Default NVM Value	Description
15:4	Reserved	0xFFF	Reserved.
3	400G-AUI8	1b	
2	400G-AUI8-AOC\ACC	1b	
1	400GBASE-DR8	1b	
0	400GBASE-LR8	1b	

6.3.42.11 Port PHY Types 6[n] (0x000A + 10*n, n=0...7)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.42.12 Port PHY Types 7[n] (0x000B + 10*n, n=0...7)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.43 RDE Ethernet MTU Section

Table 6-49. RDE Ethernet MTU Section Summary Table

Word Offset	Description	Section Reference
0x0000	RDE Ethernet MTU Type	6.3.43.1
0x0001	Length	6.3.43.2
0x0002 + 1*n, n=0...7	Ethernet MTU Size	6.3.43.3

6.3.43.1 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

6.3.43.2 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

6.3.43.3 Ethernet MTU Size[n] (0x0002 + 1*n, n=0...7)

Bits	Field Name	Default NVM Value	Description
15:0	Ethernet MTU Size	0x5DC	

6.3.44 Default DCB Parameters Section

Table 6-50. Default DCB Parameters Section Summary Table

Word Offset	Description	Section Reference
0x0000	Default DCB Parameters Type	6.3.44.1
0x0001	Length	6.3.44.2
0x0002	Ports 0-3 Mode	6.3.44.3
0x0003	Ports 4-7 Mode	6.3.44.4

6.3.44.1 Default DCB Parameters Type (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x137	Valid values are: 0x137 = Default DCB Parameters Type

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: Default DCB Parameters -> Default DCB Parameters Type Last Section -> Word: Default DCB Parameters -> Ports 4-7 Mode

6.3.44.3 Ports 0-3 Mode (0x0002)

Bits	Field Name	Default NVM Value	Description
15:12	Port 3 Mode	0x1	Default DCB Mode to apply default DCB settings after link up or not. Applies to LAN Port 3
11:8	Port 2 Mode	0x1	Default DCB Mode to apply default DCB settings after link up or not. Applies to LAN Port 2
7:4	Port 1 Mode	0x1	Default DCB Mode to apply default DCB settings after link up or not. Applies to LAN Port 1
3:0	Port 0 Mode	0x1	Default DCB Mode to apply default DCB settings after link up or not. Applies to LAN Port 0

6.3.44.4 Ports 4-7 Mode (0x0003)

Bits	Field Name	Default NVM Value	Description
15:12	Port 7 Mode	0x1	Default DCB Mode to apply default DCB settings after link up or not. Applies to LAN Port 7
11:8	Port 6 Mode	0x1	Default DCB Mode to apply default DCB settings after link up or not. Applies to LAN Port 6
7:4	Port 5 Mode	0x1	Default DCB Mode to apply default DCB settings after link up or not. Applies to LAN Port 5
3:0	Port 4 Mode	0x1	Default DCB Mode to apply default DCB settings after link up or not. Applies to LAN Port 4

6.3.45 Current DCB Parameters Section

Table 6-51. Current DCB Parameters Section Summary Table

Word Offset	Description	Section Reference
0x0000	Current DCB Parameters Type	6.3.45.1
0x0001	Length	6.3.45.2
0x0002	Ports 0-3 Mode	6.3.45.3
0x0003	Ports 4-7 Mode	6.3.45.4

6.3.45.1 Current DCB Parameters Type (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x138	Valid values are: 0x138 = Current DCB Parameters Type

6.3.45.2 Length (0x0001)

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 2 First Section -> Word: Current DCB Parameters -> Current DCB Parameters Type Last Section -> Word: Current DCB Parameters -> Ports 4-7 Mode

6.3.45.3 Ports 0-3 Mode (0x0002)

Bits	Field Name	Default NVM Value	Description
15:12	Port 3 Mode	0xF	Current DCB Mode to apply default DCB settings after link up or not. Applies to LAN Port 3
11:8	Port 2 Mode	0xF	Current DCB Mode to apply default DCB settings after link up or not. Applies to LAN Port 2
7:4	Port 1 Mode	0xF	Current DCB Mode to apply default DCB settings after link up or not. Applies to LAN Port 1
3:0	Port 0 Mode	0xF	Current DCB Mode to apply default DCB settings after link up or not. Applies to LAN Port 0

6.3.45.4 Ports 4-7 Mode (0x0003)

Bits	Field Name	Default NVM Value	Description
15:12	Port 7 Mode	0xF	Current DCB Mode to apply default DCB settings after link up or not. Applies to LAN Port 7
11:8	Port 6 Mode	0xF	Current DCB Mode to apply default DCB settings after link up or not. Applies to LAN Port 6
7:4	Port 5 Mode	0xF	Current DCB Mode to apply default DCB settings after link up or not. Applies to LAN Port 5
3:0	Port 4 Mode	0xF	Current DCB Mode to apply default DCB settings after link up or not. Applies to LAN Port 4

6.3.46 Padding Module Section

Table 6-52. Padding Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type - Padding	6.3.46.1
0x0001	Length	6.3.46.2
0x0002	Padding	6.3.46.3

6.3.46.1 Sub Module Type - Padding (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0xFFFF	Valid values are: 0xFFFF - Padding Module

6.3.46.2 Length (0x0001)

Bit(s)	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

6.3.46.3 Padding (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:0	[New Field]	0xFFFF	

6.3.47 PCIR Type 1/2 Section

Table 6-53. PCIR Type 1/2 Section Summary Table

Word Offset	Description	Section Reference
0x0000 - 0x0003	NVM contents for GLPCI_PWRDATA	6.3.47.1
0x0004 - 0x0008	NVM contents for GLPCI_PMSUP	6.3.47.2
0x0009 - 0x000A	NVM contents for GLPCI_REVID	6.3.47.3
0x000B - 0x000E	Reserved	6.3.47.4

6.3.47.1 GLPCI_PWRDATA (0x0000 - 0x00003)

6.3.47.1.1 Address Low at GLPCI_PWRDATA (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLPCI_PWRDATA	0x9DE7C	
3:0	Type	0x1	

6.3.47.1.2 Address High at GLPCI_PWRDATA (0x000`)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLPCI_PWRDATA		

6.3.47.1.3 Data Low of GLPCI_PWRDATA (0x0002)

6.3.47.1.4 Data High of GLPCI_PWRDATA (0x0003)

6.3.47.2 GLPCI_PMSUP (0x0004 - 0x0008)

6.3.47.2.1 Starting Address Low at GLPCI_PMSUP (0x0004)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLPCI_PMSUP	0x9DE94	
3:0	Type	0x2	

6.3.47.2.2 Starting Address High at GLPCI_PMSUP (0x0005)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLPCI_PMSUP		

6.3.47.2.3 Attributes at GLPCI_PMSUP (0x0006)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	00b	
2:0	Width	000b	

6.3.47.2.4 Data Low of GLPCI_PMSUP (0x0007)

6.3.47.2.5 Data High of GLPCI_PMSUP (0x0008)

6.3.47.3 GLPCI_REVID (0x0009 - 0x000A)

6.3.47.3.1 Data Low of GLPCI_REVID (0x0009)

6.3.47.3.2 Data High of GLPCI_REVID (0x000A)

6.3.47.4 Reserved (0x000B - 0x000E)

6.3.48 POR Type 1/2 Section

Table 6-54. POR Type 1/2 Section Summary Table

Word Offset	Description	Section Reference
0x0000 - 0x0021	Reserved	6.3.48.1

6.3.48.1 Reserved (0x0000 - 0x0021)

6.3.49 CORER Registers Auto-Load Module Section

Default setup to registers and internal memories that load on CORER events.

Table 6-55. CORER Registers Auto-Load Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	Module Length	6.3.49.1
0x0001 - 0x0013	NVM contents for PRT_TDPUL2TAGSEN	6.3.49.2
0x0014 - 0x0026	NVM contents for GL_SWT_L2TAGTXIB	6.3.49.3
0x0027 - 0x0039	NVM contents for GL_SWT_L2TAGRXEB	6.3.49.4
0x003A - 0x003D	NVM contents for GL_RDPU_CNTRL	6.3.49.5
0x003E - 0x0050	NVM contents for PFLAN_DB_QALLOC	6.3.49.6
0x0051 - 0x0063	NVM contents for PFLAN_CP_QALLOC	6.3.49.7
0x0064 - 0x0067	NVM contents for GLDCB_GENC	6.3.49.8
0x0068 - 0x018F	Reserved	6.3.49.9
0x0190 - 0x0194	NVM contents for GLTSYN_SYNC_DLAY	6.3.49.10
0x0195 - 0x0196	NVM contents for GLTSYN_HH_DLAY	6.3.49.11
0x0197 - 0x019B	NVM contents for GLTPB_PACING_25G	6.3.49.12
0x019C - 0x019D	NVM contents for GLTPB_PACING_10G	6.3.49.13
0x019E - 0x019F	NVM contents for GLTPB_PORT_PACING_SPEED	6.3.49.14
0x01A0 - 0x01A7	Reserved	6.3.49.15
0x01A8 - 0x01CA	NVM contents for GLRPB_DHW	6.3.49.16
0x01CB - 0x01ED	NVM contents for GLRPB_DLW	6.3.49.17
0x01EE - 0x020D	NVM contents for GLRPB_DPS	6.3.49.18
0x020E - 0x021D	NVM contents for GLRPB_SPS	6.3.49.19
0x021E - 0x0230	NVM contents for GLRPB_SHW	6.3.49.20
0x0231 - 0x0240	NVM contents for GLRPB_SLW	6.3.49.21
0x0241 - 0x0244	Reserved	6.3.49.22
0x0245 - 0x0287	NVM contents for GLRPB_TCHW	6.3.49.23
0x0288 - 0x02C7	NVM contents for GLRPB_TCLW	6.3.49.24
0x02C8 - 0x02DA	NVM contents for PRTDCB_TCUPM_REG_PE_HB_DTHR	6.3.49.25
0x02DB - 0x02ED	NVM contents for PRTDCB_TCUPM_WAIT_PFC_PE_HB_DTHR	6.3.49.26
0x02EE - 0x0330	NVM contents for TCDCB_TCUPM_WAIT_PE_HB_DTHR	6.3.49.27
0x0331 - 0x0335	NVM contents for GLDCB_TCUPM_NO_EXCEED_DIS	6.3.49.28
0x0336 - 0x0337	NVM contents for GLDCB_TCUPM_WB_DIS	6.3.49.29
0x0338 - 0x034A	NVM contents for GLHMC_PFPESDPART_FPMAT	6.3.49.30
0x034B - 0x038D	NVM contents for GLHMC_VFSDPART_FPMAT	6.3.49.31
0x038E - 0x03D0	NVM contents for GLDCB_RETSTCC	6.3.49.32
0x03D1 - 0x03E3	NVM contents for PRTDCB_RPPMC	6.3.49.33
0x03E4 - 0x03E8	NVM contents for GLDCB_RSPMC	6.3.49.34

Table 6-55. CORER Registers Auto-Load Module Section Summary Table [continued]

Word Offset	Description	Section Reference
0x03E9 - 0x03EA	NVM contents for GLDCB_RMPMC	6.3.49.35
0x03EB - 0x03F1	Reserved	6.3.49.36
0x03F2 - 0x0404	NVM contents for PFINT_TSYN_MSK	6.3.49.37
0x0405 - 0x0408	NVM contents for GLINT_CTL	6.3.49.38
0x0409 - 0x041B	NVM contents for PFGEN_PORTNUM	6.3.49.39
0x041C - 0x042E	NVM contents for PF_VT_PFALLOC	6.3.49.40
0x042F - 0x0441	NVM contents for PFLAN_RX_QALLOC	6.3.49.41
0x0442 - 0x0454	NVM contents for PFLAN_TX_QALLOC	6.3.49.42
0x0455 - 0x0467	NVM contents for PFINT_ALLOC	6.3.49.43
0x0468 - 0x047A	NVM contents for GL_SWT_L2TAGCTRL	6.3.49.44
0x047B - 0x04BD	NVM contents for GLDCB_PRS_RETSTCC	6.3.49.45
0x04BE - 0x04C1	NVM contents for GLDCB_PRS_RSPMC	6.3.49.46
0x04C2 - 0x04E4	NVM contents for GLRPRS_PMCFG_DPS	6.3.49.47
0x04E5 - 0x0507	NVM contents for GLRPRS_PMCFG_DHW	6.3.49.48
0x0508 - 0x0527	NVM contents for GLRPRS_PMCFG_DLW	6.3.49.49
0x0528 - 0x0537	NVM contents for GLRPRS_PMCFG_SPS	6.3.49.50
0x0538 - 0x054A	NVM contents for GLRPRS_PMCFG_SHW	6.3.49.51
0x054B - 0x055A	NVM contents for GLRPRS_PMCFG_SLW	6.3.49.52
0x055B - 0x059D	NVM contents for GLRPRS_PMCFG_TCHW	6.3.49.53
0x059E - 0x05DD	NVM contents for GLRPRS_PMCFG_TCLW	6.3.49.54
0x05DE - 0x05E2	NVM contents for GL_SWT_LAT_SINGLE	6.3.49.55
0x05E3 - 0x05E4	NVM contents for GL_SWT_LAT_DOUBLE	6.3.49.56
0x05E5 - 0x05E6	NVM contents for GL_SWT_LAT_QUAD	6.3.49.57
0x05E7 - 0x06FF	Reserved	6.3.49.58
0x0700 - 0x0742	NVM contents for GLDCB_SWT_RETSTCC	6.3.49.59
0x0743 - 0x074B	NVM contents for GL_PSTEXT_FORCE_PID	6.3.49.60
0x074C - 0x0754	NVM contents for GL_PREEXT_FORCE_PID	6.3.49.61
0x0755 - 0x075D	NVM contents for GL_ACLEXT_FORCE_PID	6.3.49.62
0x075E - 0x0761	NVM contents for GL_SWT_SWIDFVIDX	6.3.49.63
0x0762 - 0x0765	NVM contents for GLLAN_RCTL_1	6.3.49.64
0x0766 - 0x0778	NVM contents for GLLAN_PF_RECIPE	6.3.49.65
0x0779 - 0x09A8	NVM contents for VPDSI_TX_QTABLE_PQM	6.3.49.66
0x097C - 0x099B	NVM contents for VPLAN_DSI_VF_MODE	6.3.49.67
0x099C - 0x09BE	NVM contents for GLCOMM_QUANTA_PROF	6.3.49.68
0x09BF - 0x09CE	NVM contents for GLCOMM_PKT_SHAPER_PROF	6.3.49.69
0x09CF - 0x09DD	Reserved	6.3.49.70
0x09DE - 0x09E2	NVM contents for GL_MDCK_CFG1_TX_PQM	6.3.49.71

Table 6-55. CORER Registers Auto-Load Module Section Summary Table [continued]

Word Offset	Description	Section Reference
0x09E3 - 0x09E4	Reserved	6.3.49.72
0x09E5 - 0x09E6	NVM contents for GL_MDCK_EN_TX_PQM	6.3.49.73
0x09E7 - 0x2A2B	Reserved	6.3.49.74
0x2A2C - 0x2A2F	NVM contents for GLQF_FD_SIZE	6.3.49.75
0x2A30 - 0x2A42	NVM contents for GLHMC_PFPESDPART	6.3.49.76
0x2A43 - 0x2A46	Reserved	6.3.49.77
0x2A47 - 0x2A89	NVM contents for GLHMC_VFSDPART	6.3.49.78
0x2A8A - 0x2A8D	NVM contents for GLCOMM_MIN_MAX_PKT	6.3.49.79
0x2A8E - 0x2A8F	Reserved	6.3.49.80
0x2A90	DPU_IMEM Attributes	6.3.49.81
0x2A91 - 0x2A92	Reserved	6.3.49.82
0x2A93	DPU_IMEM Data	6.3.49.83
0x4A93 - 0x4A94	Reserved	6.3.49.84
0x4A95	DPU_RECIPE_ADDRESS Attributes	6.3.49.85
0x4A96 - 0x4A97	Reserved	6.3.49.86
0x4A98	DPU_RECIPE_ADDRESS Data	6.3.49.87
0x4C98 - 0x4C99	Reserved	6.3.49.88
0x4C9A	DPU_RECIPE_CAM Attributes	6.3.49.89
0x4C9B - 0x4C9C	Reserved	6.3.49.90
0x4C9D	DPU_RECIPE_CAM Data	6.3.49.91
0x508B - 0x509E	Reserved	6.3.49.92
0x509F	DPU_RECIPE_MASK Attributes	6.3.49.93
0x50A0 - 0x50A1	Reserved	6.3.49.94
0x50A2	DPU_RECIPE_MASK Data	6.3.49.95
0x50C2 - 0x50C3	Reserved	6.3.49.96
0x50C4	ANA_IMEM Attributes	6.3.49.97
0x50C5 - 0x50C6	Reserved	6.3.49.98
0x50C7	ANA_IMEM Data	6.3.49.99
0x5167 - 0x5168	Reserved	6.3.49.100
0x5169	ANA_NH Attributes	6.3.49.101
0x516A - 0x516B	Reserved	6.3.49.102
0x516C	ANA_NH Data	6.3.49.103
0x51BC - 0x51BD	Reserved	6.3.49.104
0x51BE	ANA_SKIP Attributes	6.3.49.105
0x51BF - 0x51C0	Reserved	6.3.49.106
0x51C1	ANA_SKIP Data	6.3.49.107
0x5211 - 0x5212	Reserved	6.3.49.108

Table 6-55. CORER Registers Auto-Load Module Section Summary Table [continued]

Word Offset	Description	Section Reference
0x5213	ANA_REPLACE Attributes	6.3.49.109
0x5214 - 0x5215	Reserved	6.3.49.110
0x5216	ANA_REPLACE Data	6.3.49.111
0x5266 - 0x5267	Reserved	6.3.49.112
0x5268	ANA_MERGE Attributes	6.3.49.113
0x5269 - 0x526A	Reserved	6.3.49.114
0x526B	ANA_MERGE Data	6.3.49.115

6.3.49.1 Module Length (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 1 First Section -> Word: CORER Registers Auto-Load Module -> Module Length Last Section -> Word: CORER Registers Auto-Load Module -> ANA_MERGE Data

6.3.49.2 PRT_TDPUL2TAGSEN (0x0001 - 0x0013)

6.3.49.2.1 Starting Address Low at PRT_TDPUL2TAGSEN (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRT_TDPUL2TAGSEN, for PRT[0]	0x40BA0	
3:0	Type	0x2	

6.3.49.2.2 Starting Address High at PRT_TDPUL2TAGSEN (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRT_TDPUL2TAGSEN, for PRT[0]		

6.3.49.2.3 Attributes at PRT_TDPUL2TAGSEN (0x0003)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.2.4 Data Low of PRT_TDPUL2TAGSEN[PRT] (0x0004 + 2*PRT, PRT=0...7)

6.3.49.2.5 Data High of PRT_TDPUL2TAGSEN[PRT] (0x0005 + 2*PRT, PRT=0...7)

6.3.49.3 GL_SWT_L2TAGTXIB (0x0014 - 0x0026)

6.3.49.3.1 Starting Address Low at GL_SWT_L2TAGTXIB (0x0014)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GL_SWT_L2TAGTXIB	0x492E8	
3:0	Type		

6.3.49.3.2 Starting Address High at GL_SWT_L2TAGTXIB (0x0015)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GL_SWT_L2TAGTXIB		

6.3.49.3.3 Attributes at GL_SWT_L2TAGTXIB (0x0016)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.3.4 Data Low of GL_SWT_L2TAGTXIB[n] (0x0017 + 2*n, n=0...7)

6.3.49.3.5 Data High of GL_SWT_L2TAGTXIB[n] (0x0018 + 2*n, n=0...7)

6.3.49.4 GL_SWT_L2TAGRXEB (0x0027 - 0x0039)

6.3.49.4.1 Starting Address Low at GL_SWT_L2TAGRXEB (0x0027)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GL_SWT_L2TAGRXEB	0x52000	
3:0	Type	0x2	

6.3.49.4.2 Starting Address High at GL_SWT_L2TAGRXEB (0x0028)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GL_SWT_L2TAGRXEB		

6.3.49.4.3 Attributes at GL_SWT_L2TAGRXEB (0x0029)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.4.4 Data Low of GL_SWT_L2TAGRXEB[n] (0x002A + 2*n, n=0...7)

6.3.49.4.5 Data High of GL_SWT_L2TAGRXEB[n] (0x002B + 2*n, n=0...7)

6.3.49.5 GL_RDPU_CNTRL (0x003A - 0x003D)

6.3.49.5.1 Address Low at GL_RDPU_CNTRL (0x003A)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GL_RDPU_CNTRL	0x52054	
3:0	Type	0x1	

6.3.49.5.2 Address High at GL_RDPU_CNTRL (0x003B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GL_RDPU_CNTRL		

6.3.49.5.3 Reserved (0x003C - 0x003D)

6.3.49.6 PFLAN_DB_QALLOC (0x003E - 0x0050)

6.3.49.6.1 Starting Address Low at PFLAN_DB_QALLOC (0x003E)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFLAN_DB_QALLOC, for PF[0]	0x75680	
3:0	Type	0x2	

6.3.49.6.2 Starting Address High at PFLAN_DB_QALLOC (0x003F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFLAN_DB_QALLOC, for PF[0]		

6.3.49.6.3 Attributes at PFLAN_DB_QALLOC (0x0040)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.6.4 Data Low of PFLAN_DB_QALLOC[PF] (0x0041 + 2*PF, PF=0...7)

6.3.49.6.5 Data High of PFLAN_DB_QALLOC[PF] (0x0042 + 2*PF, PF=0...7)

6.3.49.7 PFLAN_CP_QALLOC (0x0051 - 0x0063)

6.3.49.7.1 Starting Address Low at PFLAN_CP_QALLOC (0x0051)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFLAN_CP_QALLOC, for PF[0]	0x75700	
3:0	Type	0x2	

6.3.49.7.2 Starting Address High at PFLAN_CP_QALLOC (0x0052)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFLAN_CP_QALLOC, for PF[0]		

6.3.49.7.3 Attributes at PFLAN_CP_QALLOC (0x0053)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.7.4 Data Low of PFLAN_CP_QALLOC[PF] (0x0054 + 2*PF, PF=0...7)

6.3.49.7.5 Data High of PFLAN_CP_QALLOC[PF] (0x0055 + 2*PF, PF=0...7)

6.3.49.8 GLDCB_GENC (0x0064 - 0x0067)

6.3.49.8.1 Address Low at GLDCB_GENC (0x0064)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLDCB_GENC	0x83044	
3:0	Type	0x1	

6.3.49.8.2 Address High at GLDCB_GENC (0x0065)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLDCB_GENC		

6.3.49.8.3 Data Low of GLDCB_GENC (0x0066)

6.3.49.8.4 Data High of GLDCB_GENC (0x0067)

6.3.49.9 Reserved (0x0068 - 0x018F)

6.3.49.10 GLTSYN_SYNC_DLAY (0x0190 - 0x0194)

6.3.49.10.1 Starting Address Low at GLTSYN_SYNC_DLAY (0x0190)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLTSYN_SYNC_DLAY	0x88818	
3:0	Type	0x2	

6.3.49.10.2 Starting Address High at GLTSYN_SYNC_DLAY (0x0191)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLTSYN_SYNC_DLAY		

6.3.49.10.3 Attributes at GLTSYN_SYNC_DLAY (0x0192)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.10.4 Data Low of GLTSYN_SYNC_DLAY (0x0193)

6.3.49.10.5 Data High of GLTSYN_SYNC_DLAY (0x0194)

6.3.49.11 GLTSYN_HH_DLAY (0x0195 - 0x0196)

6.3.49.11.1 Data Low of GLTSYN_HH_DLAY (0x0195)

6.3.49.11.2 Data High of GLTSYN_HH_DLAY (0x0196)

6.3.49.12 GLTPB_PACING_25G (0x0197 - 0x019B)

6.3.49.12.1 Starting Address Low at GLTPB_PACING_25G (0x0197)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLTPB_PACING_25G	0x994E0	
3:0	Type	0x2	

6.3.49.12.2 Starting Address High at GLTPB_PACING_25G (0x0198)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLTPB_PACING_25G		

6.3.49.12.3 Attributes at GLTPB_PACING_25G (0x0199)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x3	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.12.4 Data Low of GLTPB_PACING_25G (0x019A)

6.3.49.12.5 Data High of GLTPB_PACING_25G (0x019B)

6.3.49.13 GLTPB_PACING_10G (0x019C - 0x019D)

6.3.49.13.1 Data Low of GLTPB_PACING_10G (0x019C)

6.3.49.13.2 Data High of GLTPB_PACING_10G (0x019D)

6.3.49.14 GLTPB_PORT_PACING_SPEED (0x019E - 0x019F)

6.3.49.14.1 Data Low of GLTPB_PORT_PACING_SPEED (0x019E)

6.3.49.14.2 Data High of GLTPB_PORT_PACING_SPEED (0x019F)

6.3.49.15 Reserved (0x01A0 - 0x01A7)

6.3.49.16 GLRPB_DHW (0x01A8 - 0x01CA)

6.3.49.16.1 Starting Address Low at GLRPB_DHW (0x01A8)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLRPB_DHW	0xAC000	
3:0	Type	0x2	

6.3.49.16.2 Starting Address High at GLRPB_DHW (0x01A9)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLRPB_DHW		

6.3.49.16.3 Attributes at GLRPB_DHW (0x01AA)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x10	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.16.4 Data Low of GLRPB_DHW[n] (0x01AB + 2*n, n=0...15)

6.3.49.16.5 Data High of GLRPB_DHW[n] (0x01AC + 2*n, n=0...15)

6.3.49.17 GLRPB_DLW (0x01CB - 0x01ED)

6.3.49.17.1 Starting Address Low at GLRPB_DLW (0x01CB)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLRPB_DLW	0xAC044	
3:0	Type	0x2	

6.3.49.17.2 Starting Address High at GLRPB_DLW (0x01CC)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLRPB_DLW		

6.3.49.17.3 Attributes at GLRPB_DLW (0x01CD)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x28	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.17.4 Data Low of GLRPB_DLW[n] (0x01CE + 2*n, n=0...15)

6.3.49.17.5 Data High of GLRPB_DLW[n] (0x01CF + 2*n, n=0...15)

6.3.49.18 GLRPB_DPS (0x01EE - 0x020D)

6.3.49.18.1 Data Low of GLRPB_DPS[n] (0x01EE + 2*n, n=0...15)

6.3.49.18.2 Data High of GLRPB_DPS[n] (0x01EF + 2*n, n=0...15)

6.3.49.19 GLRPB_SPS (0x020E - 0x021D)

6.3.49.19.1 Data Low of GLRPB_SPS[n] (0x020E + 2*n, n=0...7)

6.3.49.19.2 Data High of GLRPB_SPS[n] (0x020F + 2*n, n=0...7)

6.3.49.20 GLRPB_SHW (0x021E - 0x0230)

6.3.49.20.1 Starting Address Low at GLRPB_SHW (0x021E)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLRPB_SHW	0xAC120	
3:0	Type	0x2	

6.3.49.20.2 Starting Address High at GLRPB_SHW (0x021F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLRPB_SHW		

6.3.49.20.3 Attributes at GLRPB_SHW (0x0220)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x10	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.20.4 Data Low of GLRPB_SHW[n] (0x0221 + 2*n, n=0...7)

6.3.49.20.5 Data High of GLRPB_SHW[n] (0x0222 + 2*n, n=0...7)

6.3.49.21 GLRPB_SLW (0x0231 - 0x0240)

6.3.49.21.1 Data Low of GLRPB_SLW[n] (0x0231 + 2*n, n=0...7)

6.3.49.21.2 Data High of GLRPB_SLW[n] (0x0232 + 2*n, n=0...7)

6.3.49.22 Reserved (0x0241 - 0x0244)

6.3.49.23 GLRPB_TCHW (0x0245 - 0x0287)

6.3.49.23.1 Starting Address Low at GLRPB_TCHW (0x0245)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLRPB_TCHW	0xAC330	
3:0	Type	0x2	

6.3.49.23.2 Starting Address High at GLRPB_TCHW (0x0246)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLRPB_TCHW		

6.3.49.23.3 Attributes at GLRPB_TCHW (0x0247)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x40	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.23.4 Data Low of GLRPB_TCHW[n] (0x0248 + 2*n, n=0...31)

6.3.49.23.5 Data High of GLRPB_TCHW[n] (0x0249 + 2*n, n=0...31)

6.3.49.24 GLRPB_TCLW (0x0288 - 0x02C7)

6.3.49.24.1 Data Low of GLRPB_TCLW[n] (0x0288 + 2*n, n=0...31)

6.3.49.24.2 Data High of GLRPB_TCLW[n] (0x0289 + 2*n, n=0...31)

6.3.49.25 PRTDCB_TCUPM_REG_PE_HB_DTHR (0x02C8 - 0x02DA)

6.3.49.25.1 Starting Address Low at PRTDCB_TCUPM_REG_PE_HB_DTHR (0x02C8)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTDCB_TCUPM_REG_PE_HB_DTHR, for PRT[0]	0xBC420	
3:0	Type	0x2	

6.3.49.25.2 Starting Address High at PRTDCB_TCUPM_REG_PE_HB_DTHR (0x02C9)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTDCB_TCUPM_REG_PE_HB_DTHR, for PRT[0]		

6.3.49.25.3 Attributes at PRTDCB_TCUPM_REG_PE_HB_DTHR (0x02CA)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.25.4 Data Low of PRTDCB_TCUPM_REG_PE_HB_DTHR[PRT] (0x02CB + 2*PRT, PRT=0...7)

6.3.49.25.5 Data High of PRTDCB_TCUPM_REG_PE_HB_DTHR[PRT] (0x02CC + 2*PRT, PRT=0...7)

6.3.49.26 PRTDCB_TCUPM_WAIT_PFC_PE_HB_DTHR (0x02DB - 0x02ED)

6.3.49.26.1 Starting Address Low at PRTDCB_TCUPM_WAIT_PFC_PE_HB_DTHR (0x02DB)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTDCB_TCUPM_WAIT_PFC_PE_HB_DTHR, for PRT[0]	0xBC4E0	
3:0	Type	0x2	

6.3.49.26.2 Starting Address High at PRTDCB_TCUPM_WAIT_PFC_PE_HB_DTHR (0x02DC)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTDCB_TCUPM_WAIT_PFC_PE_HB_DTHR, for PRT[0]		

6.3.49.26.3 Attributes at PRTDCB_TCUPM_WAIT_PFC_PE_HB_DTHR (0x02DD)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.26.4 Data Low of PRTDCB_TCUPM_WAIT_PFC_PE_HB_DTHR[PRT] (0x02DE + 2*PRT, PRT=0...7)

6.3.49.26.5 Data High of PRTDCB_TCUPM_WAIT_PFC_PE_HB_DTHR[PRT] (0x02DF + 2*PRT, PRT=0...7)

6.3.49.27 TCDCB_TCUPM_WAIT_PE_HB_DTHR (0x02EE - 0x0330)

6.3.49.27.1 Starting Address Low at TCDCB_TCUPM_WAIT_PE_HB_DTHR (0x02EE)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of TCDCB_TCUPM_WAIT_PE_HB_DTHR	0xBC7A0	
3:0	Type	0x2	

6.3.49.27.2 Starting Address High at TCDCB_TCUPM_WAIT_PE_HB_DTHR (0x02EF)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of TCDCB_TCUPM_WAIT_PE_HB_DTHR		

6.3.49.27.3 Attributes at TCDCB_TCUPM_WAIT_PE_HB_DTHR (0x02F0)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x20	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.27.4 Data Low of TCDCB_TCUPM_WAIT_PE_HB_DTHR[n] (0x02F1 + 2*n, n=0...31)

6.3.49.27.5 Data High of TCDCB_TCUPM_WAIT_PE_HB_DTHR[n] (0x02F2 + 2*n, n=0...31)

6.3.49.28 GLDCB_TCUPM_NO_EXCEED_DIS (0x0331 - 0x0335)

6.3.49.28.1 Starting Address Low at GLDCB_TCUPM_NO_EXCEED_DIS (0x0331)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLDCB_TCUPM_NO_EXCEED_DIS	0xBC830	
3:0	Type	0x2	

6.3.49.28.2 Starting Address High at GLDCB_TCUPM_NO_EXCEED_DIS (0x0332)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLDCB_TCUPM_NO_EXCEED_DIS		

6.3.49.28.3 Attributes at GLDCB_TCUPM_NO_EXCEED_DIS (0x0333)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.28.4 Data Low of GLDCB_TCUPM_NO_EXCEED_DIS (0x0334)

6.3.49.28.5 Data High of GLDCB_TCUPM_NO_EXCEED_DIS (0x0335)

6.3.49.29 GLDCB_TCUPM_WB_DIS (0x0336 - 0x0337)

6.3.49.29.1 Data Low of GLDCB_TCUPM_WB_DIS (0x0336)

6.3.49.29.2 Data High of GLDCB_TCUPM_WB_DIS (0x0337)

6.3.49.30 GLHMC_PFPESDPART_FPMAT (0x0338 - 0x034A)

6.3.49.30.1 Starting Address Low at GLHMC_PFPESDPART_FPMAT (0x0338)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLHMC_PFPESDPART_FPMAT	0x100880	
3:0	Type	0x2	

6.3.49.30.2 Starting Address High at GLHMC_PFPESDPART_FPMAT (0x0339)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLHMC_PFPESDPART_FPMAT		

6.3.49.30.3 Attributes at GLHMC_PFPESDPART_FPMAT (0x033A)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.30.4 Data Low of GLHMC_PFPESDPART_FPMAT[n] (0x033B + 2*n, n=0...7)

6.3.49.30.5 Data High of GLHMC_PFPESDPART_FPMAT[n] (0x033C + 2*n, n=0...7)

6.3.49.31 GLHMC_VFSDPART_FPMAT (0x034B - 0x038D)

6.3.49.31.1 Starting Address Low at GLHMC_VFSDPART_FPMAT (0x034B)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLHMC_VFSDPART_FPMAT	0x108800	
3:0	Type	0x2	

6.3.49.31.2 Starting Address High at GLHMC_VFSDPART_FPMAT (0x034C)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLHMC_VFSDPART_FPMAT		

6.3.49.31.3 Attributes at GLHMC_VFSDPART_FPMAT (0x034D)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x20	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.31.4 Data Low of GLHMC_VFSDPART_FPMAT[n] (0x034E + 2*n, n=0...31)

6.3.49.31.5 Data High of GLHMC_VFSDPART_FPMAT[n] (0x034F + 2*n, n=0...31)

6.3.49.32 GLDCB_RETSTCC (0x038E - 0x03D0)

6.3.49.32.1 Starting Address Low at GLDCB_RETSTCC (0x038E)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLDCB_RETSTCC	0x122140	
3:0	Type	0x2	

6.3.49.32.2 Starting Address High at GLDCB_RETSTCC (0x038F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLDCB_RETSTCC		

6.3.49.32.3 Attributes at GLDCB_RETSTCC (0x0390)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x20	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.32.4 Data Low of GLDCB_RETSTCC[n] (0x0391 + 2*n, n=0...31)

6.3.49.32.5 Data High of GLDCB_RETSTCC[n] (0x0392 + 2*n, n=0...31)

6.3.49.33 PRTDCB_RPPMC (0x03D1 - 0x03E3)

6.3.49.33.1 Starting Address Low at PRTDCB_RPPMC (0x03D1)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTDCB_RPPMC, for PRT[0]	0x122240	
3:0	Type	0x2	

6.3.49.33.2 Starting Address High at PRTDCB_RPPMC (0x03D2)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTDCB_RPPMC, for PRT[0]		

6.3.49.33.3 Attributes at PRTDCB_RPPMC (0x03D3)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.33.4 Data Low of PRTDCB_RPPMC[PRT] (0x03D4 + 2*PRT, PRT=0)

6.3.49.33.5 Data High of PRTDCB_RPPMC[PRT] (0x03D5 + 2*PRT, PRT=0)

6.3.49.34 GLDCB_RSPMC (0x03E4 - 0x03E8)

6.3.49.34.1 Starting Address Low at GLDCB_RSPMC (0x03E4)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLDCB_RSPMC	0x1223C4	
3:0	Type	0x2	

6.3.49.34.2 Starting Address High at GLDCB_RSPMC (0x03E5)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLDCB_RSPMC		

6.3.49.34.3 Attributes at GLDCB_RSPMC (0x03E6)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x2	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.34.4 Data Low of GLDCB_RSPMC (0x03E7)

6.3.49.34.5 Data High of GLDCB_RSPMC (0x03E8)

6.3.49.35 GLDCB_RMPMC (0x03E9 - 0x03EA)

6.3.49.35.1 Data Low of GLDCB_RMPMC (0x03E9)

6.3.49.35.2 Data High of GLDCB_RMPMC (0x03EA)

6.3.49.36 Reserved (0x03EB - 0x03F1)

6.3.49.37 PFINT_TSYN_MSK (0x03F2 - 0x0404)

6.3.49.37.1 Starting Address Low at PFINT_TSYN_MSK (0x03F2)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFINT_TSYN_MSK, for PF[0]	0x16C980	
3:0	Type		

6.3.49.37.2 Starting Address High at PFINT_TSYN_MSK (0x03F3)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFINT_TSYN_MSK, for PF[0]		

6.3.49.37.3 Attributes at PFINT_TSYN_MSK (0x03F4)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.37.4 Data Low of PFINT_TSYN_MSK[PF] (0x03F5 + 2*PF, PF=0...7)

6.3.49.37.5 Data High of PFINT_TSYN_MSK[PF] (0x03F6 + 2*PF, PF=0...7)

6.3.49.38 GLINT_CTL (0x0405 - 0x0408)

6.3.49.38.1 Address Low at GLINT_CTL (0x0405)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLINT_CTL	0x16CC54	
3:0	Type	0x1	

6.3.49.38.2 Address High at GLINT_CTL (0x0406)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLINT_CTL		

6.3.49.38.3 Data Low of GLINT_CTL (0x0407)

6.3.49.38.4 Data High of GLINT_CTL (0x0408)

6.3.49.39 PFGEN_PORTNUM (0x0409 - 0x041B)

6.3.49.39.1 Starting Address Low at PFGEN_PORTNUM (0x0409)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFGEN_PORTNUM, for PF[0]	0x1D2400	
3:0	Type	0x2	

6.3.49.39.2 Starting Address High at PFGEN_PORTNUM (0x040A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFGEN_PORTNUM, for PF[0]		

6.3.49.39.3 Attributes at PFGEN_PORTNUM (0x040B)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.39.4 Data Low of PFGEN_PORTNUM[PF] (0x040C + 2*PF, PF=0...7)

6.3.49.39.5 Data High of PFGEN_PORTNUM[PF] (0x040D + 2*PF, PF=0...7)

6.3.49.40 PF_VT_PFALLOC (0x041C - 0x042E)

6.3.49.40.1 Starting Address Low at PF_VT_PFALLOC (0x041C)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PF_VT_PFALLOC, for PF[0]	0x1D2480	
3:0	Type	0x2	

6.3.49.40.2 Starting Address High at PF_VT_PFALLOC (0x041D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PF_VT_PFALLOC, for PF[0]		

6.3.49.40.3 Attributes at PF_VT_PFALLOC (0x041E)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.40.4 Data Low of PF_VT_PFALLOC[PF] (0x041F + 2*PF, PF=0...7)

6.3.49.40.5 Data High of PF_VT_PFALLOC[PF] (0x0420 + 2*PF, PF=0...7)

6.3.49.41 PFLAN_RX_QALLOC (0x042F - 0x0441)

6.3.49.41.1 Starting Address Low at PFLAN_RX_QALLOC (0x042F)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFLAN_RX_QALLOC, for PF[0]	0x1D2500	
3:0	Type	0x2	

6.3.49.41.2 Starting Address High at PFLAN_RX_QALLOC (0x0430)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFLAN_RX_QALLOC, for PF[0]		

6.3.49.41.3 Attributes at PFLAN_RX_QALLOC (0x0431)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.41.4 Data Low of PFLAN_RX_QALLOC[PF] (0x0432 + 2*PF, PF=0...7)

6.3.49.41.5 Data High of PFLAN_RX_QALLOC[PF] (0x0433 + 2*PF, PF=0...7)

6.3.49.42 PFLAN_TX_QALLOC (0x0442 - 0x0454)

6.3.49.42.1 Starting Address Low at PFLAN_TX_QALLOC (0x0442)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFLAN_TX_QALLOC, for PF[0]	0x1D2580	
3:0	Type	0x2	

6.3.49.42.2 Starting Address High at PFLAN_TX_QALLOC (0x0443)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFLAN_TX_QALLOC, for PF[0]		

6.3.49.42.3 Attributes at PFLAN_TX_QALLOC (0x0444)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.42.4 Data Low of PFLAN_TX_QALLOC[PF] (0x0445 + 2*PF, PF=0...7)

6.3.49.42.5 Data High of PFLAN_TX_QALLOC[PF] (0x0446 + 2*PF, PF=0...7)

6.3.49.43 PFINT_ALLOC (0x0455 - 0x0467)

6.3.49.43.1 Starting Address Low at PFINT_ALLOC (0x0455)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFINT_ALLOC, for PF[0]	0x1D2600	
3:0	Type	0x2	

6.3.49.43.2 Starting Address High at PFINT_ALLOC (0x0456)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFINT_ALLOC, for PF[0]		

6.3.49.43.3 Attributes at PFINT_ALLOC (0x0457)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.43.4 Data Low of PFINT_ALLOC[PF] (0x0458 + 2*PF, PF=0...7)

6.3.49.43.5 Data High of PFINT_ALLOC[PF] (0x0459 + 2*PF, PF=0...7)

6.3.49.44 GL_SWT_L2TAGCTRL (0x0468 - 0x047A)

6.3.49.44.1 Starting Address Low at GL_SWT_L2TAGCTRL (0x0468)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GL_SWT_L2TAGCTRL	0x1D2660	
3:0	Type	0x2	

6.3.49.44.2 Starting Address High at GL_SWT_L2TAGCTRL (0x0469)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GL_SWT_L2TAGCTRL		

6.3.49.44.3 Attributes at GL_SWT_L2TAGCTRL (0x046A)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.44.4 Data Low of GL_SWT_L2TAGCTRL[n] (0x046B + 2*n, n=0...7)

6.3.49.44.5 Data High of GL_SWT_L2TAGCTRL[n] (0x046C + 2*n, n=0...7)

6.3.49.45 GLDCB_PRS_RETSTCC (0x047B - 0x04BD)

6.3.49.45.1 Starting Address Low at GLDCB_PRS_RETSTCC (0x047B)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLDCB_PRS_RETSTCC	0x2000B0	
3:0	Type	0x2	

6.3.49.45.2 Starting Address High at GLDCB_PRS_RETSTCC (0x047C)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLDCB_PRS_RETSTCC		

6.3.49.45.3 Attributes at GLDCB_PRS_RETSTCC (0x047D)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x20	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.45.4 Data Low of GLDCB_PRS_RETSTCC[n] (0x047E + 2*n, n=0...31)

6.3.49.45.5 Data High of GLDCB_PRS_RETSTCC[n] (0x047F + 2*n, n=0...31)

6.3.49.46 GLDCB_PRS_RSPMC (0x04BE - 0x04C1)

6.3.49.46.1 Address Low at GLDCB_PRS_RSPMC (0x04BE)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLDCB_PRS_RSPMC	0x200160	
3:0	Type	0x1	

6.3.49.46.2 Address High at GLDCB_PRS_RSPMC (0x04BF)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLDCB_PRS_RSPMC		

6.3.49.46.3 Data Low of GLDCB_PRS_RSPMC (0x04C0)

6.3.49.46.4 Data High of GLDCB_PRS_RSPMC (0x04C1)

6.3.49.47 GLRPRS_PMCFG_DPS (0x04C2 - 0x04E4)

6.3.49.47.1 Starting Address Low at GLRPRS_PMCFG_DPS (0x04C2)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLRPRS_PMCFG_DPS	0x200308	
3:0	Type	0x2	

6.3.49.47.2 Starting Address High at GLRPRS_PMCFG_DPS (0x04C3)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLRPRS_PMCFG_DPS		

6.3.49.47.3 Attributes at GLRPRS_PMCFG_DPS (0x04C4)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x10	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.47.4 Data Low of GLRPRS_PMCFG_DPS[n] (0x04C5 + 2*n, n=0...15)

6.3.49.47.5 Data High of GLRPRS_PMCFG_DPS[n] (0x04C6 + 2*n, n=0...15)

6.3.49.48 GLRPRS_PMCFG_DHW (0x04E5 - 0x0507)

6.3.49.48.1 Starting Address Low at GLRPRS_PMCFG_DHW (0x04E5)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLRPRS_PMCFG_DHW	0x200388	
3:0	Type	0x2	

6.3.49.48.2 Starting Address High at GLRPRS_PMCFG_DHW (0x04E6)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLRPRS_PMCFG_DHW		

6.3.49.48.3 Attributes at GLRPRS_PMCFG_DHW (0x04E7)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x28	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.48.4 Data Low of GLRPRS_PMCFG_DHW[n] (0x04E8 + 2*n, n=0...15)

6.3.49.48.5 Data High of GLRPRS_PMCFG_DHW[n] (0x04E9 + 2*n, n=0...15)

6.3.49.49 GLRPRS_PMCFG_DLW (0x0508 - 0x0527)

6.3.49.49.1 Data Low of GLRPRS_PMCFG_DLW[n] (0x0508 + 2*n, n=0...15)

6.3.49.49.2 Data High of GLRPRS_PMCFG_DLW[n] (0x0509 + 2*n, n=0...15)

6.3.49.50 GLRPRS_PMCFG_SPS (0x0528 - 0x0537)

6.3.49.50.1 Data Low of GLRPRS_PMCFG_SPS[n] (0x0528 + 2*n, n=0...7)

6.3.49.50.2 Data High of GLRPRS_PMCFG_SPS[n] (0x0529 + 2*n, n=0...7)

6.3.49.51 GLRPRS_PMCFG_SHW (0x0538 - 0x054A)

6.3.49.51.1 Starting Address Low at GLRPRS_PMCFG_SHW (0x0538)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLRPRS_PMCFG_SHW	0x200448	
3:0	Type	0x2	

6.3.49.51.2 Starting Address High at GLRPRS_PMCFG_SHW (0x0539)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLRPRS_PMCFG_SHW		

6.3.49.51.3 Attributes at GLRPRS_PMCFG_SHW (0x053A)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x10	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.51.4 Data Low of GLRPRS_PMCFG_SHW[n] (0x053B + 2*n, n=0...7)

6.3.49.51.5 Data High of GLRPRS_PMCFG_SHW[n] (0x053C + 2*n, n=0...7)

6.3.49.52 GLRPRS_PMCFG_SLW (0x054B - 0x055A)

6.3.49.52.1 Data Low of GLRPRS_PMCFG_SLW[n] (0x054B + 2*n, n=0...7)

6.3.49.52.2 Data High of GLRPRS_PMCFG_SLW[n] (0x054C + 2*n, n=0...7)

6.3.49.53 GLRPRS_PMCFG_TCHW (0x055B - 0x059D)

6.3.49.53.1 Starting Address Low at GLRPRS_PMCFG_TCHW (0x055B)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLRPRS_PMCFG_TCHW	0x200588	
3:0	Type	0x2	

6.3.49.53.2 Starting Address High at GLRPRS_PMCFG_TCHW (0x055C)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLRPRS_PMCFG_TCHW		

6.3.49.53.3 Attributes at GLRPRS_PMCFG_TCHW (0x055D)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x40	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.53.4 Data Low of GLRPRS_PMCFG_TCHW[n] (0x055E + 2*n, n=0...31)

6.3.49.53.5 Data High of GLRPRS_PMCFG_TCHW[n] (0x055F + 2*n, n=0...31)

6.3.49.54 GLRPRS_PMCFG_TCLW (0x059E - 0x05DD)

6.3.49.54.1 Data Low of GLRPRS_PMCFG_TCLW[n] (0x059E + 2*n, n=0...31)

6.3.49.54.2 Data High of GLRPRS_PMCFG_TCLW[n] (0x059F + 2*n, n=0...31)

6.3.49.55 GL_SWT_LAT_SINGLE (0x05DE - 0x05E2)

6.3.49.55.1 Starting Address Low at GL_SWT_LAT_SINGLE (0x05DE)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GL_SWT_LAT_SINGLE	0x204000	
3:0	Type	0x2	

6.3.49.55.2 Starting Address High at GL_SWT_LAT_SINGLE (0x05DF)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GL_SWT_LAT_SINGLE		

6.3.49.55.3 Attributes at GL_SWT_LAT_SINGLE (0x05E0)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x3	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.55.4 Data Low of GL_SWT_LAT_SINGLE (0x05E1)

6.3.49.55.5 Data High of GL_SWT_LAT_SINGLE (0x05E2)

6.3.49.56 GL_SWT_LAT_DOUBLE (0x05E3 - 0x05E4)

6.3.49.56.1 Data Low of GL_SWT_LAT_DOUBLE (0x05E3)

6.3.49.56.2 Data High of GL_SWT_LAT_DOUBLE (0x05E4)

6.3.49.57 GL_SWT_LAT_QUAD (0x05E5 - 0x05E6)

6.3.49.57.1 Data Low of GL_SWT_LAT_QUAD (0x05E5)

6.3.49.57.2 Data High of GL_SWT_LAT_QUAD (0x05E6)

6.3.49.58 Reserved (0x05E7 - 0x06FF)

6.3.49.59 GLDCB_SWT_RETSTCC (0x0700 - 0x0742)

6.3.49.59.1 Starting Address Low at GLDCB_SWT_RETSTCC (0x0700)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLDCB_SWT_RETSTCC	0x20A040	
3:0	Type	0x2	

6.3.49.59.2 Starting Address High at GLDCB_SWT_RETSTCC (0x0701)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLDCB_SWT_RETSTCC		

6.3.49.59.3 Attributes at GLDCB_SWT_RETSTCC (0x0702)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x20	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.59.4 Data Low of GLDCB_SWT_RETSTCC[n] (0x0703 + 2*n, n=0...31)

6.3.49.59.5 Data High of GLDCB_SWT_RETSTCC[n] (0x0704 + 2*n, n=0...31)

6.3.49.60 GL_PSTEXT_FORCE_PID (0x0743 - 0x074B)

6.3.49.60.1 Starting Address Low at GL_PSTEXT_FORCE_PID (0x0743)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GL_PSTEXT_FORCE_PID	0x20E000	
3:0	Type	0x2	

6.3.49.60.2 Starting Address High at GL_PSTEXT_FORCE_PID (0x0744)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GL_PSTEXT_FORCE_PID		

6.3.49.60.3 Attributes at GL_PSTEXT_FORCE_PID (0x0745)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x3	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.60.4 Data Low of GL_PSTEXT_FORCE_PID[n] (0x0746 + 2*n, n=0...2)

6.3.49.60.5 Data High of GL_PSTEXT_FORCE_PID[n] (0x0747 + 2*n, n=0...2)

6.3.49.61 GL_PREEXT_FORCE_PID (0x074C - 0x0754)

6.3.49.61.1 Starting Address Low at GL_PREEXT_FORCE_PID (0x074C)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GL_PREEXT_FORCE_PID	0x20F000	
3:0	Type	0x2	

6.3.49.61.2 Starting Address High at GL_PREEXT_FORCE_PID (0x074D)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GL_PREEXT_FORCE_PID		

6.3.49.61.3 Attributes at GL_PREEXT_FORCE_PID (0x074E)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x3	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.61.4 Data Low of GL_PREEXT_FORCE_PID[n] (0x074F + 2*n, n=0...2)

6.3.49.61.5 Data High of GL_PREEXT_FORCE_PID[n] (0x0750 +2*n, n=0...2)

6.3.49.62 GL_ACLEXT_FORCE_PID (0x0755 - 0x075D)

6.3.49.62.1 Starting Address Low at GL_ACLEXT_FORCE_PID (0x0755)

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GL_ACLEXT_FORCE_PID	0x210000	
3:0	Type	0x2	

6.3.49.62.2 Starting Address High at GL_ACLEXT_FORCE_PID (0x0756)

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GL_ACLEXT_FORCE_PID		

6.3.49.62.3 Attributes at GL_ACLEXT_FORCE_PID (0x0757)

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x3	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.62.4 Data Low of GL_ACLEXT_FORCE_PID[n] (0x0758 + 2*n, n=0...2)

6.3.49.62.5 Data High of GL_ACLEXT_FORCE_PID[n] (0x0759 + 2*n, n=0...2)

6.3.49.63 GL_SWT_SWIDFVIDX (0x075E - 0x0761)

6.3.49.63.1 Address Low at GL_SWT_SWIDFVIDX (0x075E)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GL_SWT_SWIDFVIDX	0x214114	
3:0	Type	0x1	

6.3.49.63.2 Address High at GL_SWT_SWIDFVIDX (0x075F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GL_SWT_SWIDFVIDX		

6.3.49.63.3 Data Low of GL_SWT_SWIDFVIDX (0x0760)

6.3.49.63.4 Data High of GL_SWT_SWIDFVIDX (0x0761)

6.3.49.64 GLLAN_RCTL_1 (0x0762 - 0x0765)

6.3.49.64.1 Address Low at GLLAN_RCTL_1 (0x0762)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLLAN_RCTL_1	0x2941FC	
3:0	Type	0x1	

6.3.49.64.2 Address High at GLLAN_RCTL_1 (0x0763)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLLAN_RCTL_1		

6.3.49.64.3 Data Low of GLLAN_RCTL_1 (0x0764)

6.3.49.64.4 Data High of GLLAN_RCTL_1 (0x0765)

6.3.49.65 GLLAN_PF_RECIPE (0x0766 - 0x0778)

6.3.49.65.1 Starting Address Low at GLLAN_PF_RECIPE (0x0766)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLLAN_PF_RECIPE	0x29420C	
3:0	Type	0x2	

6.3.49.65.2 Starting Address High at GLLAN_PF_RECIPE (0x0767)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLLAN_PF_RECIPE		

6.3.49.65.3 Attributes at GLLAN_PF_RECIPE (0x0768)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.65.4 Data Low of GLLAN_PF_RECIPE[n] (0x0769 + 2*n, n=0...7)

6.3.49.65.5 Data High of GLLAN_PF_RECIPE[n] (0x076A + 2*n, n=0...7)

6.3.49.66 VPDSI_TX_QTABLE_PQM (0x0779 - 0x09A8)

6.3.49.66.1 Starting Address Low at VPDSI_TX_QTABLE_PQM (0x0779)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of VPDSI_TX_QTABLE_PQM, for VP16[0]	0x2D2800	
3:0	Type	0x2	

6.3.49.66.2 Starting Address High at VPDSI_TX_QTABLE_PQM (0x077A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of VPDSI_TX_QTABLE_PQM, for VP16[0]		

6.3.49.66.3 Attributes at VPDSI_TX_QTABLE_PQM (0x077B)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x110	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.67 VPLAN_DSI_VF_MODE (0x097C - 0x099B)

6.3.49.67.1 Data Low of VPLAN_DSI_VF_MODE[VP16] (0x097C + 2*VP16, VP16=0...15)

6.3.49.67.2 Data High of VPLAN_DSI_VF_MODE[VP16] (0x097D + 2*VP16, VP16=0...15)

6.3.49.68 GLCOMM_QUANTA_PROF (0x099C - 0x09BE)

6.3.49.68.1 Starting Address Low at GLCOMM_QUANTA_PROF (0x099C)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLCOMM_QUANTA_PROF	0x2D2D68	
3:0	Type	0x2	

6.3.49.68.2 Starting Address High at GLCOMM_QUANTA_PROF (0x099D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLCOMM_QUANTA_PROF		

6.3.49.68.3 Attributes at GLCOMM_QUANTA_PROF (0x099E)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x18	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.68.4 Data Low of GLCOMM_QUANTA_PROF[n] (0x099F + 2*n, n=0...15)

6.3.49.68.5 Data High of GLCOMM_QUANTA_PROF[n] (0x09A0+ 2*n, n=0...15)

6.3.49.69 GLCOMM_PKT_SHAPER_PROF (0x09BF - 0x09CE)

6.3.49.69.1 Data Low of GLCOMM_PKT_SHAPER_PROF[n] (0x09BF + 2*n, n=0...7)

6.3.49.69.2 Data High of GLCOMM_PKT_SHAPER_PROF[n] (0x09C0 + 2*n, n=0...7)

6.3.49.70 Reserved (0x09CF - 0x09DD)

6.3.49.71 GL_MDCK_CFG1_TX_PQM (0x09DE - 0x09E2)

6.3.49.71.1 Starting Address Low at GL_MDCK_CFG1_TX_PQM (0x09DE)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GL_MDCK_CFG1_TX_PQM	0x2D2DF4	
3:0	Type	0x2	

6.3.49.71.2 Starting Address High at GL_MDCK_CFG1_TX_PQM (0x09DF)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GL_MDCK_CFG1_TX_PQM		

6.3.49.71.3 Attributes at GL_MDCK_CFG1_TX_PQM (0x09E0)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x3	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.71.4 Data Low of GL_MDCK_CFG1_TX_PQM (0x09E1)

6.3.49.71.5 Data High of GL_MDCK_CFG1_TX_PQM (0x09E2)

6.3.49.72 Reserved (0x09E3 - 0x09E4)

6.3.49.73 GL_MDCK_EN_TX_PQM (0x09E5 - 0x09E6)

6.3.49.73.1 Data Low of GL_MDCK_EN_TX_PQM (0x09E5)

6.3.49.73.2 Data High of GL_MDCK_EN_TX_PQM (0x09E6)

6.3.49.74 Reserved (0x0E7 - 0x2A2B)

6.3.49.75 GLQF_FD_SIZE (0x2A2C - 0x2A2F)

6.3.49.75.1 Address Low at GLQF_FD_SIZE (0x2A2C)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLQF_FD_SIZE	0x460010	
3:0	Type	0x1	

6.3.49.75.2 Address High at GLQF_FD_SIZE (0x2A2D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLQF_FD_SIZE		

6.3.49.75.3 Data Low of GLQF_FD_SIZE (0x2A2E)

6.3.49.75.4 Data High of GLQF_FD_SIZE (0x2A2F)

6.3.49.76 GLHMC_PFPESDPART (0x2A30 - 0x2A42)

6.3.49.76.1 Starting Address Low at GLHMC_PFPESDPART (0x2A30)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLHMC_PFPESDPART	0x520880	
3:0	Type	0x2	

6.3.49.76.2 Starting Address High at GLHMC_PFPESDPART (0x2A41)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLHMC_PFPESDPART		

6.3.49.76.3 Attributes at GLHMC_PFPESDPART (0x2A32)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.76.4 Data Low of GLHMC_PFPESDPART[n] (0x2A33 + 2*n, n=0...7)

6.3.49.76.5 Data High of GLHMC_PFPESDPART[n] (0x2A34 + 2*n, n=0...7)

6.3.49.77 Reserved (0x2A43 - 0x2A46)

6.3.49.78 GLHMC_VFSDPART (0x2A47 - 0x2A89)

6.3.49.78.1 Starting Address Low at GLHMC_VFSDPART (0x2A47)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLHMC_VFSDPART	0x528800	
3:0	Type	0x2	

6.3.49.78.2 Starting Address High at GLHMC_VFSDPART (0x2A48)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLHMC_VFSDPART		

6.3.49.78.3 Attributes at GLHMC_VFSDPART (0x2A49)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x20	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.78.4 Data Low of GLHMC_VFSDPART[n] (0x2A4A + 2*n, n=0...31)

6.3.49.78.5 Data High of GLHMC_VFSDPART[n] (0x2A4B + 2*n, n=0...31)

6.3.49.79 GLCOMM_MIN_MAX_PKT (0x2A8A - 0x2A8D)

6.3.49.79.1 Address Low at GLCOMM_MIN_MAX_PKT (0x2A8A)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLCOMM_MIN_MAX_PKT	0xFC064	
3:0	Type	0x1	

6.3.49.79.2 Address High at GLCOMM_MIN_MAX_PKT (0x2A8B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLCOMM_MIN_MAX_PKT		

6.3.49.79.3 Data Low of GLCOMM_MIN_MAX_PKT (0x2A8C)

6.3.49.79.4 Data High of GLCOMM_MIN_MAX_PKT (0x2A8D)

6.3.49.80 Reserved (0x2A8E - 0x2A8F)

6.3.49.81 DPU_IMEM Attributes (0x2A90)

Part of a Type 3 structure to load the DPU_IMEM.

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x400	
4:3	Skip	00b	
2:0	Width	010b	

6.3.49.82 Reserved (0x2A91 - 0x2A92)

6.3.49.83 DPU_IMEM Data (0x2A93)

Raw data module length: 8192 words.

Part of a Type 3 structure to load the DPU_IMEM.

6.3.49.84 Reserved (0x4A93 - 0x4A94)

6.3.49.85 DPU_RECIPE_ADDRESS Attributes (0x4A95)

Part of a Type 3 structure to load the DPU_RECIPE_ADDRESS.

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x100	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.86 Reserved (0x4A96 - 0x4A97)

6.3.49.87 DPU_RECIPE_ADDRESS Data (0x4A98)

Raw data module length: 512 words.

Part of a Type 3 structure to load the DPU_RECIPE_ADDRESS.

6.3.49.88 Reserved (0x4C98 - 0x4C99)

6.3.49.89 DPU_RECIPE_CAM Attributes (0x4C9A)

Part of a Type 3 structure to load the DPU_RECIPE_CAM.

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x100	
4:3	Skip	00b	
2:0	Width	001b	

6.3.49.90 Reserved (0x4C9B - 0x4C9C)

6.3.49.91 DPU_RECIPE_CAM Data (0x4C9D)

Raw data module length: 1024 words.

Part of a Type 3 structure to load the DPU_RECIPE_CAM.

6.3.49.92 Reserved (0x509D - 0x509E)

6.3.49.93 DPU_RECIPE_MASK Attributes (0x509F)

Part of a Type 3 structure to load the DPU_RECIPE_MASK.

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x008	
4:3	Skip	00b	
2:0	Width	001b	

6.3.49.94 Reserved (0x50A0 - 0x50A1)

6.3.49.95 DPU_RECIPE_MASK Data (0x509A2)

Raw data module length: 32 words.

Part of a Type 3 structure to load the DPU_RECIPE_MASK.

6.3.49.96 Reserved (0x50C2 - 0x50C3)

6.3.49.97 ANA_IMEM Attributes (0x50C4)

Part of a Type 3 structure to load the DPU_IMEM.

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x28	
4:3	Skip	00b	
2:0	Width	001b	

6.3.49.98 Reserved (0x50C5 - 0x50C6)

6.3.49.99 ANA_IMEM Data (0x50C7)

Raw data module length: 160 words.

Part of a Type 3 structure to load the DPU_IMEM.

6.3.49.100 Reserved (0x5167 - 0x5168)

6.3.49.101 ANA_NH Attributes (0x5169)

Part of a Type 3 structure to load the DPU_IMEM.

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x28	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.102 Reserved (0x516A - 0x516B)

6.3.49.103 ANA_NH Data (0x516C)

Raw data module length: 80 words.

Part of a Type 3 structure to load the DPU_IMEM.

6.3.49.104 Reserved (0x51BC - 0x51BD)

6.3.49.105 ANA_SKIP Attributes (0x51BE)

Part of a Type 3 structure to load the DPU_IMEM.

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x28	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.106 Reserved (0x51BF - 0x51C0)

6.3.49.107 ANA_SKIP Data (0x51C1)

Raw data module length: 80 words.

Part of a Type 3 structure to load the DPU_IMEM.

6.3.49.108 Reserved (0x5211 - 0x5212)

6.3.49.109 ANA_REPLACE Attributes (0x5213)

Part of a Type 3 structure to load the DPU_IMEM.

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x28	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.110 Reserved (0x5214 - 0x5215)

6.3.49.111 ANA_REPLACE Data (0x5216)

Raw data module length: 80 words.

Part of a Type 3 structure to load the DPU_IMEM.

6.3.49.112 Reserved (0x5266 - 0x5267)

6.3.49.113 ANA_MERGE Attributes (0x5268)

Part of a Type 3 structure to load the DPU_IMEM.

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x28	
4:3	Skip	00b	
2:0	Width	000b	

6.3.49.114 Reserved (0x5269 - 0x526A)

6.3.49.115 ANA_MERGE Data (0x526B)

Raw data module length: 80 words.

Part of a Type 3 structure to load the DPU_IMEM.

6.3.50 GLOBR Registers Auto-Load Module Section

Default setup to registers that load on GLOBR events.

Table 6-56. GLOBR Registers Auto-Load Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	Module Length	6.3.50.1
0x0001 - 0x0004	NVM contents for GLGEN_MAC_LINK_TOPO	6.3.50.2
0x0005 - 0x0017	Reserved	6.3.50.3
0x0018 - 0x002A	NVM contents for PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE	6.3.50.4
0x002B - 0x003A	NVM contents for PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE	6.3.50.5
0x003B - 0x004A	NVM contents for PRTMAC_HSEC_CTL_RX_ENABLE_GCP	6.3.50.6
0x004B - 0x005D	NVM contents for PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GCP	6.3.50.7
0x005E - 0x006D	NVM contents for PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART1	6.3.50.8
0x006E - 0x007D	NVM contents for PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART2	6.3.50.9
0x007E - 0x00E6	Reserved	6.3.50.10
0x00E7 - 0x00F9	NVM contents for PRTMAC_HSEC_CTL_RX_ENABLE_GPP	6.3.50.11
0x00FA - 0x010C	Reserved	6.3.50.12
0x010D - 0x011F	NVM contents for PRTMAC_HSEC_CTL_RX_ENABLE_PPP	6.3.50.13
0x0120 - 0x0132	Reserved	6.3.50.14
0x0133 - 0x0145	NVM contents for PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL	6.3.50.15
0x0146 - 0x01D5	NVM contents for PRTMAC_HSEC_CTL_TX_PAUSE_QUANTA	6.3.50.16
0x01D6 - 0x0265	NVM contents for PRTMAC_HSEC_CTL_TX_PAUSE_REFRESH_TIMER	6.3.50.17
0x0266 - 0x0278	NVM contents for PRTMAC_HSEC_CTL_TX_SA_PART1	6.3.50.18
0x0279 - 0x0288	NVM contents for PRTMAC_HSEC_CTL_TX_SA_PART2	6.3.50.19
0x0289 - 0x02E6	Reserved	6.3.50.20
0x02E7 - 0x02F9	NVM contents for PRTPM_EEER	6.3.50.21
0x02FA - 0x0309	NVM contents for PRTPM_EEEC	6.3.50.22
0x030A - 0x0355	NVM contents for PRTDCB_FCTTVN	6.3.50.23
0x034D - 0x035C	NVM contents for PRTDCB_FCRTV	6.3.50.24
0x035D - 0x036F	NVM contents for PRTDCB_FCCFG	6.3.50.25
0x0370 - 0x039F	Reserved	6.3.50.26
0x03A0	Attributes at PRTGEN_CNF2, for PRT[0]	6.3.50.27
0x03A1	Data Low of PRTGEN_CNF2, for PRT[0]	6.3.50.28
0x03A2	Data High of PRTGEN_CNF2, for PRT[0]	6.3.50.29
0x03A3	Data Low of PRTGEN_CNF2, for PRT[1]	6.3.50.30
0x03A4	Data High of PRTGEN_CNF2, for PRT[1]	6.3.50.31
0x03A5	Data Low of PRTGEN_CNF2, for PRT[2]	6.3.50.32
0x03A6	Data High of PRTGEN_CNF2, for PRT[2]	6.3.50.33
0x03A7	Data Low of PRTGEN_CNF2, for PRT[3]	6.3.50.34

Table 6-56. GLOBR Registers Auto-Load Module Section Summary Table [continued]

Word Offset	Description	Section Reference
0x03A8	Data High of PRTGEN_CNF2, for PRT[3]	6.3.50.35
0x03A9	Data Low of PRTGEN_CNF2, for PRT[4]	6.3.50.36
0x03AA	Data High of PRTGEN_CNF2, for PRT[4]	6.3.50.37
0x03AB	Data Low of PRTGEN_CNF2, for PRT[5]	6.3.50.38
0x03AC	Data High of PRTGEN_CNF2, for PRT[5]	6.3.50.39
0x03AD	Data Low of PRTGEN_CNF2, for PRT[6]	6.3.50.40
0x03AE	Data High of PRTGEN_CNF2, for PRT[6]	6.3.50.41
0x03AF	Data Low of PRTGEN_CNF2, for PRT[7]	6.3.50.42
0x03B0	Data High of PRTGEN_CNF2, for PRT[7]	6.3.50.43

6.3.50.1 Module Length (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 1 First Section -> Word: GLOBR Registers Auto-load Module -> Module Length Last Section -> Word: GLOBR Registers Auto-load Module -> Data High of PRTGEN_CNF2, for PRT[7]

6.3.50.2 GLGEN_MAC_LINK_TOPO (0x0001 - 0x0004)

6.3.50.2.1 Address Low at GLGEN_MAC_LINK_TOPO (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLGEN_MAC_LINK_TOPO	0xB81DC	
3:0	Type	0x1	

6.3.50.2.2 Address High at GLGEN_MAC_LINK_TOPO (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLGEN_MAC_LINK_TOPO		

6.3.50.2.3 Data Low of GLGEN_MAC_LINK_TOPO (0x0003)

6.3.50.2.4 Data High of GLGEN_MAC_LINK_TOPO (0x0004)

6.3.50.3 Reserved (0x0005 - 0x0017)

6.3.50.4 PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE (0x0018 - 0x002A)

6.3.50.4.1 Starting Address Low at PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE (0x0018)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE, for PRT[0]	0x1E3180	
3:0	Type	0x2	

6.3.50.4.2 Starting Address High at PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE (0x0019)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE, for PRT[0]		

6.3.50.4.3 Attributes at PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE (0x001A)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x18	
4:3	Skip	00b	
2:0	Width	000b	

6.3.50.4.4 Data Low of PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE[PRT] (0x001B + 2*PRT, PRT=0...7)

6.3.50.4.5 Data High of PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE[PRT] (0x001C + 2*PRT, PRT=0...7)

6.3.50.5 PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE (0x002B - 0x003A)

6.3.50.5.1 Data Low of PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE[PRT] (0x002B + 2*PRT, PRT=0...7)

6.3.50.5.2 Data High of PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE[PRT] (0x002C + 2*PRT, PRT=0...7)

6.3.50.6 PRTMAC_HSEC_CTL_RX_ENABLE_GCP (0x003B - 0x004A)

6.3.50.6.1 Data Low of PRTMAC_HSEC_CTL_RX_ENABLE_GCP[PRT] (0x003B + 2*PRT, PRT=0...7)

6.3.50.6.2 Data High of PRTMAC_HSEC_CTL_RX_ENABLE_GCP[PRT] (0x003C + 2*PRT, PRT=0...7)

6.3.50.7 PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GCP (0x004B - 0x005D)

6.3.50.7.1 Starting Address Low at PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GCP (0x004B)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GCP, for PRT[0]	0x1E3200	
3:0	Type	0x2	

6.3.50.7.2 Starting Address High at PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GCP (0x004C)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GCP, for PRT[0]		

6.3.50.7.3 Attributes at PRTMAC_HSEC_CTL_RX_CHECK_UCAST_GCP (0x004D)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x20	
4:3	Skip	00b	
2:0	Width	000b	

6.3.50.7.4 Reserved (0x004E - 0x005D)

6.3.50.8 PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART 1 (0x005E - 0x006D)

6.3.50.8.1 Data Low of PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART1[PRT] (0x005E + 2*PRT, PRT=0...7)

6.3.50.8.2 Data High of PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART1[PRT] (0x005F + 2*PRT, PRT=0...7)

6.3.50.9 PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART 2 (0x006E - 0x007D)

6.3.50.9.1 Data Low of PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART2[PRT] (0x006E + 2*PRT, PRT=0...7)

6.3.50.9.2 Data High of PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART2[PRT] (0x006F + 2*PRT, PRT=0...7)

6.3.50.10 Reserved (0x007E - 0x00E6)

6.3.50.11 PRTMAC_HSEC_CTL_RX_ENABLE_GPP (0x00E7 - 0x00F9)

6.3.50.11.1 Starting Address Low at PRTMAC_HSEC_CTL_RX_ENABLE_GPP (0x00E7)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_RX_ENABLE_GPP, for PRT[0]	0x1E34C0	
3:0	Type	0x2	

6.3.50.11.2 Starting Address High at PRTMAC_HSEC_CTL_RX_ENABLE_GPP (0x00E8)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_RX_ENABLE_GPP, for PRT[0]		

6.3.50.11.3 Attributes at PRTMAC_HSEC_CTL_RX_ENABLE_GPP (0x00E9)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.50.11.4 Data Low of PRTMAC_HSEC_CTL_RX_ENABLE_GPP[PRT] (0x00EA + 2*PRT, PRT=0...7)

6.3.50.11.5 Data High of PRTMAC_HSEC_CTL_RX_ENABLE_GPP[PRT] (0x00EB + 2*PRT, PRT=0...7)

6.3.50.12 Reserved (0x00FA - 0x010C)

6.3.50.13 PRTMAC_HSEC_CTL_RX_ENABLE_PPP (0x010D - 0x011F)

6.3.50.13.1 Starting Address Low at PRTMAC_HSEC_CTL_RX_ENABLE_PPP (0x010D)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_RX_ENABLE_PPP, for PRT[0]	0x1E35C0	
3:0	Type	0x2	

6.3.50.13.2 Starting Address High at PRTMAC_HSEC_CTL_RX_ENABLE_PPP (0x010E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_RX_ENABLE_PPP, for PRT[0]		

6.3.50.13.3 Attributes at PRTMAC_HSEC_CTL_RX_ENABLE_PPP (0x010F)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.50.13.4 Data Low of PRTMAC_HSEC_CTL_RX_ENABLE_PPP[PRT] (0x0110 + 2*PRT, PRT=0...7)

6.3.50.13.5 Data High of PRTMAC_HSEC_CTL_RX_ENABLE_PPP[PRT] (0x0111 + 2*PRT, PRT=0...7)

6.3.50.14 Reserved (0x0120 - 0x0132)

6.3.50.15 PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL (0x0133 - 0x0145)

6.3.50.15.1 Starting Address Low at PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL (0x0133)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL, for PRT[0]	0x1E36C0	
3:0	Type	0x2	

6.3.50.15.2 Starting Address High at PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL (0x0134)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL, for PRT[0]		

6.3.50.15.3 Attributes at PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL 0x0135)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x98	
4:3	Skip	00b	
2:0	Width	000b	

6.3.50.15.4 Data Low of PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL[PRT] (0x0136 + 2*PRT, PRT=0...7)

6.3.50.15.5 Data High of PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL[PRT] (0x0137 + 2*PRT, PRT=0...7)

6.3.50.16 PRTMAC_HSEC_CTL_TX_PAUSE_QUANTA (0x0146 - 0x01D5)

6.3.50.16.1 Data Low of PRTMAC_HSEC_CTL_TX_PAUSE_QUANTA[n,PRT] (0x0146 + 16*n + 2*PRT, n=0...8, PRT=0...7)

6.3.50.16.2 Data High of PRTMAC_HSEC_CTL_TX_PAUSE_QUANTA[n,PRT] (0x0147 + 16*n + 2*PRT, n=0...8, PRT=0...7)

6.3.50.17 PRTMAC_HSEC_CTL_TX_PAUSE_REFRESH_TIMER (0x01D6 - 0x0265)

6.3.50.17.1 Data Low of PRTMAC_HSEC_CTL_TX_PAUSE_REFRESH_TIMER[n,PRT] (0x01D6 + 16*n + 2*PRT, n=0...8, PRT=0...7)

6.3.50.17.2 Data High of PRTMAC_HSEC_CTL_TX_PAUSE_REFRESH_TIMER[n,PRT] (0x01D7 + 16*n + 2*PRT, n=0...8, PRT=0...7)

6.3.50.18 PRTMAC_HSEC_CTL_TX_SA_PART1 (0x0266 - 0x0278)

6.3.50.18.1 Starting Address Low at PRTMAC_HSEC_CTL_TX_SA_PART1 (0x0266)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTMAC_HSEC_CTL_TX_SA_PART1, for PRT[0]	0x1E3960	
3:0	Type	0x2	

6.3.50.18.2 Starting Address High at PRTMAC_HSEC_CTL_TX_SA_PART1 (0x0267)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTMAC_HSEC_CTL_TX_SA_PART1, for PRT[0]		

6.3.50.18.3 Attributes at PRTMAC_HSEC_CTL_TX_SA_PART1 (0x0268)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x10	
4:3	Skip	00b	
2:0	Width	000b	

6.3.50.18.4 Data Low of PRTMAC_HSEC_CTL_TX_SA_PART1[PRT] (0x0269 + 2*PRT, PRT=0...7)

6.3.50.18.5 Data High of PRTMAC_HSEC_CTL_TX_SA_PART1[PRT] (0x026A + 2*PRT, PRT=0...7)

6.3.50.19 PRTMAC_HSEC_CTL_TX_SA_PART2 (0x0279 - 0x0288)

6.3.50.19.1 Data Low of PRTMAC_HSEC_CTL_TX_SA_PART2[PRT] (0x0279 + 2*PRT, PRT=0...7)

6.3.50.19.2 Data High of PRTMAC_HSEC_CTL_TX_SA_PART2[PRT] (0x027A + 2*PRT, PRT=0...7)

6.3.50.20 Reserved (0x0289 - 0x02E6)

6.3.50.21 PRTPM_EEER (0x02E7 - 0x02F9)

6.3.50.21.1 Starting Address Low at PRTPM_EEER (0x02E7)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTPM_EEER, for PRT[0]	0x1E4360	
3:0	Type	0x2	

6.3.50.21.2 Starting Address High at PRTPM_EEER (0x02E8)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTPM_EEER, for PRT[0]		

6.3.50.21.3 Attributes at PRTPM_EEER (0x02E9)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x10	
4:3	Skip	00b	
2:0	Width	000b	

6.3.50.21.4 Data Low of PRTPM_EEER[PRT] (0x02EA + 2*PRT, PRT=0...7)

6.3.50.21.5 Data High of PRTPM_EEER[PRT] (0x02EB + 2*PRT, PRT=0...7)

6.3.50.22 PRTPM_EEEC (0x02FA - 0x0309)

6.3.50.22.1 Data Low of PRTPM_EEEC[PRT] (0x02FA + 2*PRT, PRT=0...7)

6.3.50.22.2 Data High of PRTPM_EEEC[PRT] (0x02FB + 2*PRT, PRT=0...7)

6.3.50.23 PRTDCB_FCTTVN (0x030A - 0x0355)

6.3.50.23.1 Starting Address Low at PRTDCB_FCTTVN (0x030A)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTDCB_FCTTVN, for PRT[0]	0x1E4580	
3:0	Type	0x2	

6.3.50.23.2 Starting Address High at PRTDCB_FCTTVN (0x030B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTDCB_FCTTVN, for PRT[0]		

6.3.50.23.3 Attributes at PRTDCB_FCTTVN (0x030C)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x28	
4:3	Skip	00b	
2:0	Width	000b	

6.3.50.23.4 Data Low of PRTDCB_FCTTVN[n,PRT] (0x030D + 19*n + 2*PRT, n=0...3, PRT=0...7)

6.3.50.23.5 Data High of PRTDCB_FCTTVN[n,PRT] (0x030E + 19*n + 2*PRT, n=0...3, PRT=0...7)

6.3.50.24 PRTDCB_FCRTV (0x034D - 0x035C)

6.3.50.24.1 Data Low of PRTDCB_FCRTV[PRT] (0x034D + 2*PRT, PRT=0...7)

6.3.50.24.2 Data High of PRTDCB_FCRTV[PRT] (0x034E + 2*PRT, PRT=0...7)

6.3.50.25 PRTDCB_FCCFG (0x035D - 0x036F)

6.3.50.25.1 Starting Address Low at PRTDCB_FCCFG (0x035D)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTDCB_FCCFG, for PRT[0]	0x1E4640	
3:0	Type	0x2	

6.3.50.25.2 Starting Address High at PRTDCB_FCCFG (0x035E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTDCB_FCCFG, for PRT[0]		

6.3.50.25.3 Attributes at PRTDCB_FCCFG (0x035F)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.50.25.4 Data Low of PRTDCB_FCCFG[PRT] (0x0360 + 2*PRT, PRT=0...7)

6.3.50.25.5 Data High of PRTDCB_FCCFG[PRT] (0x0361 + 2*PRT, PRT=0...7)

6.3.50.26 Reserved (0x0370 - 0x039F)

6.3.50.27 Attributes at PRTGEN_CNF2, for PRT[0] (0x03A0)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x8	
4:3	Skip	00b	
2:0	Width	000b	

6.3.50.28 Data Low of PRTGEN_CNF2, for PRT[0] (0x03A1)

This register contains configuration per Ethernet port loaded from NVM.

Bits	Field Name	Default NVM Value	Description
15:1	Reserved	0x0	Reserved.
0	ACTIVATE_PORT_LINK	1b	When this field is set to 0b, the port's link is powered down. This field can be used by an application to disable the link until the software device driver is loaded and enables the link. ***Notes:*** 1. The PCIe functions associated with the port are not affected by the link loss. 2. Deactivating the link using this configuration is ignored when the interface is used for manageability.

6.3.50.29 Data High of PRTGEN_CNF2, for PRT[0] (0x03A2)

This register contains configuration per Ethernet port loaded from NVM.

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Reserved.

6.3.50.30 Data Low of PRTGEN_CNF2, for PRT[1] (0x03A3)

This register contains configuration per Ethernet port loaded from NVM.

Bits	Field Name	Default NVM Value	Description
15:1	Reserved	0x0	Reserved.
0	ACTIVATE_PORT_LINK	1b	When this field is set to 0b, the port's link is powered down. This field can be used by an application to disable the link until the software device driver is loaded and enables the link. ***Notes:*** 1. The PCIe functions associated with the port are not affected by the link loss. 2. Deactivating the link using this configuration is ignored when the interface is used for manageability.

6.3.50.31 Data High of PRTGEN_CNF2, for PRT[1] (0x03A4)

This register contains configuration per Ethernet port loaded from NVM.

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Reserved.

6.3.50.32 Data Low of PRTGEN_CNF2, for PRT[2] (0x03A5)

This register contains configuration per Ethernet port loaded from NVM.

Bits	Field Name	Default NVM Value	Description
15:1	Reserved	0x0	Reserved.
0	ACTIVATE_PORT_LINK	1b	When this field is set to 0b, the port's link is powered down. This field can be used by an application to disable the link until the software device driver is loaded and enables the link. ***Notes:*** 1. The PCIe functions associated with the port are not affected by the link loss. 2. Deactivating the link using this configuration is ignored when the interface is used for manageability.

6.3.50.33 Data High of PRTGEN_CNF2, for PRT[2] (0x03A6)

This register contains configuration per Ethernet port loaded from NVM.

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Reserved.

6.3.50.34 Data Low of PRTGEN_CNF2, for PRT[3] (0x03A7)

This register contains configuration per Ethernet port loaded from NVM.

Bits	Field Name	Default NVM Value	Description
15:1	Reserved	0x0	Reserved.
0	ACTIVATE_PORT_LINK	1b	When this field is set to 0b, the port's link is powered down. This field can be used by an application to disable the link until the software device driver is loaded and enables the link. ***Notes:*** 1. The PCIe functions associated with the port are not affected by the link loss. 2. Deactivating the link using this configuration is ignored when the interface is used for manageability.

6.3.50.35 Data High of PRTGEN_CNF2, for PRT[3] (0x03A8)

This register contains configuration per Ethernet port loaded from NVM.

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Reserved.

6.3.50.36 Data Low of PRTGEN_CNF2, for PRT[4] (0x03A9)

This register contains configuration per Ethernet port loaded from NVM.

Bits	Field Name	Default NVM Value	Description
15:1	Reserved	0x0	Reserved.
0	ACTIVATE_PORT_LINK	1b	When this field is set to 0b, the port's link is powered down. This field can be used by an application to disable the link until the software device driver is loaded and enables the link. ***Notes:*** 1. The PCIe functions associated with the port are not affected by the link loss. 2. Deactivating the link using this configuration is ignored when the interface is used for manageability.

6.3.50.37 Data High of PRTGEN_CNF2, for PRT[4] (0x03AA)

This register contains configuration per Ethernet port loaded from NVM.

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Reserved.

6.3.50.38 Data Low of PRTGEN_CNF2, for PRT[5] (0x03AB)

This register contains configuration per Ethernet port loaded from NVM.

Bits	Field Name	Default NVM Value	Description
15:1	Reserved	0x0	Reserved.
0	ACTIVATE_PORT_LINK	1b	When this field is set to 0b, the port's link is powered down. This field can be used by an application to disable the link until the software device driver is loaded and enables the link. ***Notes:*** 1. The PCIe functions associated with the port are not affected by the link loss. 2. Deactivating the link using this configuration is ignored when the interface is used for manageability.

6.3.50.39 Data High of PRTGEN_CNF2, for PRT[5] (0x03AC)

This register contains configuration per Ethernet port loaded from NVM.

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Reserved.

6.3.50.40 Data Low of PRTGEN_CNF2, for PRT[6] (0x03AD)

This register contains configuration per Ethernet port loaded from NVM.

Bits	Field Name	Default NVM Value	Description
15:1	Reserved	0x0	Reserved.
0	ACTIVATE_PORT_LINK	1b	When this field is set to 0b, the port's link is powered down. This field can be used by an application to disable the link until the software device driver is loaded and enables the link. ***Notes:*** 1. The PCIe functions associated with the port are not affected by the link loss. 2. Deactivating the link using this configuration is ignored when the interface is used for manageability.

6.3.50.41 Data High of PRTGEN_CNF2, for PRT[6] (0x03AE)

This register contains configuration per Ethernet port loaded from NVM.

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Reserved.

6.3.50.42 Data Low of PRTGEN_CNF2, for PRT[7] (0x03AF)

This register contains configuration per Ethernet port loaded from NVM.

Bits	Field Name	Default NVM Value	Description
15:1	Reserved	0x0	Reserved.
0	ACTIVATE_PORT_LINK	1b	When this field is set to 0b, the port's link is powered down. This field can be used by an application to disable the link until the software device driver is loaded and enables the link. ***Notes:*** 1. The PCIe functions associated with the port are not affected by the link loss. 2. Deactivating the link using this configuration is ignored when the interface is used for manageability.

6.3.50.43 Data High of PRTGEN_CNF2, for PRT[7] (0x03B0)

This register contains configuration per Ethernet port loaded from NVM.

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Reserved.

6.3.51 PE CORER Registers Section

Default setup to registers and internal memories that load on CORER events for PE.

Table 6-57. PE CORER Registers Section Summary Table

Word Offset	Description	Section Reference
0x0000	Module Length	6.3.51.1
0x0001 - 0x0017	Reserved	6.3.51.2

6.3.51.1 Module Length (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 1 First Section -> Word: PE CORER Registers -> Module Length Last Section -> Word: PE CORER Registers -> Reserved

6.3.51.2 Reserved (0x0001 - 0x0017)

6.3.52 Sideband Bus Auto-Load Section

Table 6-58. Sideband Bus Auto-Load Section Summary Table

Word Offset	Description	Section Reference
0x0000	Module Length	6.3.52.1
0x0001	CDF CFIO West - Type F Word 0	6.3.52.2
0x0002	CDF CFIO West - Type F Word 1	6.3.52.3
0x0003	CDF CFIO West - Type F Word 2	6.3.52.4
0x0004	GBE_SDP_TIMESYNC0 Address Low	6.3.52.5
0x0005	GBE_SDP_TIMESYNC0 Address High	6.3.52.6
0x0006	GBE_SDP_TIMESYNC0 Data Low	6.3.52.7
0x0007	GBE_SDP_TIMESYNC0 Data High	6.3.52.8
0x0008	GBE_SDP_TIMESYNC1 Address Low	6.3.52.9
0x0009	GBE_SDP_TIMESYNC1 Address High	6.3.52.10
0x000A	GBE_SDP_TIMESYNC1 Data Low	6.3.52.11
0x000B	GBE_SDP_TIMESYNC1 Data High	6.3.52.12
0x000C	GBE_SDP_TIMESYNC2 Address Low	6.3.52.13
0x000D	GBE_SDP_TIMESYNC2 Address High	6.3.52.14
0x000E	GBE_SDP_TIMESYNC2 Data Low	6.3.52.15
0x000F	GBE_SDP_TIMESYNC2 Data High	6.3.52.16
0x0010	GBE_SDP_TIMESYNC3 Address Low	6.3.52.17
0x0011	GBE_SDP_TIMESYNC3 Address High	6.3.52.18
0x0012	GBE_SDP_TIMESYNC3 Data Low	6.3.52.19
0x0013	GBE_SDP_TIMESYNC3 Data High	6.3.52.20
0x0014	GBE0_I2C_CLK Address Low	6.3.52.21
0x0015	GBE0_I2C_CLK Address High	6.3.52.22
0x0016	GBE0_I2C_CLK Data Low	6.3.52.23
0x0017	GBE0_I2C_CLK Data High	6.3.52.24
0x0018	GBE0_I2C_DATA Address Low	6.3.52.25
0x0019	GBE0_I2C_DATA Address High	6.3.52.26
0x001A	GBE0_I2C_DATA Data Low	6.3.52.27
0x001B	GBE0_I2C_DATA Data High	6.3.52.28
0x001C	GBE1_I2C_CLK Address Low	6.3.52.29
0x001D	GBE1_I2C_CLK Address High	6.3.52.30
0x001E	GBE1_I2C_CLK Data Low	6.3.52.31
0x001F	GBE1_I2C_CLK Data High	6.3.52.32
0x0020	GBE1_I2C_DATA Address Low	6.3.52.33
0x0021	GBE1_I2C_DATA Address High	6.3.52.34
0x0022	GBE1_I2C_DATA Data Low	6.3.52.35

Table 6-58. Sideband Bus Auto-Load Section Summary Table [continued]

Word Offset	Description	Section Reference
0x0023	GBE1_I2C_DATA Data High	6.3.52.36
0x0024	GBE2_I2C_CLK Address Low	6.3.52.37
0x0025	GBE2_I2C_CLK Address High	6.3.52.38
0x0026	GBE2_I2C_CLK Data Low	6.3.52.39
0x0027	GBE2_I2C_CLK Data High	6.3.52.40
0x0028	GBE2_I2C_DATA Address Low	6.3.52.41
0x0029	GBE2_I2C_DATA Address High	6.3.52.42
0x002A	GBE2_I2C_DATA Data Low	6.3.52.43
0x002B	GBE2_I2C_DATA Data High	6.3.52.44
0x002C	GBE3_I2C_CLK Address Low	6.3.52.45
0x002D	GBE3_I2C_CLK Address High	6.3.52.46
0x002E	GBE3_I2C_CLK Data Low	6.3.52.47
0x002F	GBE3_I2C_CLK Data High	6.3.52.48
0x0030	GBE3_I2C_DATA Address Low	6.3.52.49
0x0031	GBE3_I2C_DATA Address High	6.3.52.50
0x0032	GBE3_I2C_DATA Data Low	6.3.52.51
0x0033	GBE3_I2C_DATA Data High	6.3.52.52
0x0034	GBE0_LED0 Address Low	6.3.52.53
0x0035	GBE0_LED0 Address High	6.3.52.54
0x0036	GBE0_LED0 Data Low	6.3.52.55
0x0037	GBE0_LED0 Data High	6.3.52.56
0x0038	GBE0_LED1 Address Low	6.3.52.57
0x0039	GBE0_LED1 Address High	6.3.52.58
0x003A	GBE0_LED1 Data Low	6.3.52.59
0x003B	GBE0_LED1 Data High	6.3.52.60
0x003C	GBE0_LED2 Address Low	6.3.52.61
0x003D	GBE0_LED2 Address High	6.3.52.62
0x003E	GBE0_LED2 Data Low	6.3.52.63
0x003F	GBE0_LED2 Data High	6.3.52.64
0x0040	GBE1_LED0 Address Low	6.3.52.65
0x0041	GBE1_LED0 Address High	6.3.52.66
0x0042	GBE1_LED0 Data Low	6.3.52.67
0x0043	GBE1_LED0 Data High	6.3.52.68
0x0044	GBE1_LED1 Address Low	6.3.52.69
0x0045	GBE1_LED1 Address High	6.3.52.70
0x0046	GBE1_LED1 Data Low	6.3.52.71
0x0047	GBE1_LED1 Data High	6.3.52.72

Table 6-58. Sideband Bus Auto-Load Section Summary Table [continued]

Word Offset	Description	Section Reference
0x0048	GBE1_LED2 Address Low	6.3.52.73
0x0049	GBE1_LED2 Address High	6.3.52.74
0x004A	GBE1_LED2 Data Low	6.3.52.75
0x004B	GBE1_LED2 Data High	6.3.52.76
0x004C	GBE2_LED0 Address Low	6.3.52.77
0x004D	GBE2_LED0 Address High	6.3.52.78
0x004E	GBE2_LED0 Data Low	6.3.52.79
0x004F	GBE2_LED0 Data High	6.3.52.80
0x0050	GBE2_LED1 Address Low	6.3.52.81
0x0051	GBE2_LED1 Address High	6.3.52.82
0x0052	GBE2_LED1 Data Low	6.3.52.83
0x0053	GBE2_LED1 Data High	6.3.52.84
0x0054	GBE2_LED2 Address Low	6.3.52.85
0x0055	GBE2_LED2 Address High	6.3.52.86
0x0056	GBE2_LED2 Data Low	6.3.52.87
0x0057	GBE2_LED2 Data High	6.3.52.88
0x0058	GBE3_LED0 Address Low	6.3.52.89
0x0059	GBE3_LED0 Address High	6.3.52.90
0x005A	GBE3_LED0 Data Low	6.3.52.91
0x005B	GBE3_LED0 Data High	6.3.52.92
0x005C	GBE3_LED1 Address Low	6.3.52.93
0x005D	GBE3_LED1 Address High	6.3.52.94
0x005E	GBE3_LED1 Data Low	6.3.52.95
0x005F	GBE3_LED1 Data High	6.3.52.96
0x0060	GBE3_LED2 Address Low	6.3.52.97
0x0061	GBE3_LED2 Address High	6.3.52.98
0x0062	GBE3_LED2 Data Low	6.3.52.99
0x0063	GBE3_LED2 Data High	6.3.52.100
0x0064	GBE_SMB_CLK Address Low	6.3.52.101
0x0065	GBE_SMB_CLK Address High	6.3.52.102
0x0066	GBE_SMB_CLK Data Low	6.3.52.103
0x0067	GBE_SMB_CLK Data High	6.3.52.104
0x0068	GBE_SMB_DATA Address Low	6.3.52.105
0x0069	GBE_SMB_DATA Address High	6.3.52.106
0x006A	GBE_SMB_DATA Data Low	6.3.52.107
0x006B	GBE_SMB_DATA Data High	6.3.52.108
0x006C	GBE_SMB_ALRT_N Address Low	6.3.52.109

Table 6-58. Sideband Bus Auto-Load Section Summary Table [continued]

Word Offset	Description	Section Reference
0x006D	GBE_SMB_ALRT_N Address High	6.3.52.110
0x006E	GBE_SMB_ALRT_N Data Low	6.3.52.111
0x006F	GBE_SMB_ALRT_N Data High	6.3.52.112
0x0070	UART1_RXD Address Low	6.3.52.113
0x0071	UART1_RXD Address High	6.3.52.114
0x0072	UART1_RXD Data Low	6.3.52.115
0x0073	UART1_RXD Data High	6.3.52.116
0x0074	UART1_TXD Address Low	6.3.52.117
0x0075	UART1_TXD Address High	6.3.52.118
0x0076	UART1_TXD Data Low	6.3.52.119
0x0077	UART1_TXD Data High	6.3.52.120
0x0078	CPU_GP_0 Address Low	6.3.52.121
0x0079	CPU_GP_0 Address High	6.3.52.122
0x007A	CPU_GP_0 Data Low	6.3.52.123
0x007B	CPU_GP_0 Data High	6.3.52.124
0x007C	GBE_MNG_I2C_CLK Address Low	6.3.52.125
0x007D	GBE_MNG_I2C_CLK Address High	6.3.52.126
0x007E	GBE_MNG_I2C_CLK Data Low	6.3.52.127
0x007F	GBE_MNG_I2C_CLK Data High	6.3.52.128
0x0080	GBE_MNG_I2C_DATA Address Low	6.3.52.129
0x0081	GBE_MNG_I2C_DATA Address High	6.3.52.130
0x0082	GBE_MNG_I2C_DATA Data Low	6.3.52.131
0x0083	GBE_MNG_I2C_DATA Data High	6.3.52.132
0x0084	NCSI_RXD0 Address Low	6.3.52.133
0x0085	NCSI_RXD0 Address High	6.3.52.134
0x0086	NCSI_RXD0 Data Low	6.3.52.135
0x0087	NCSI_RXD0 Data High	6.3.52.136
0x0088	NCSI_RXD1 Address Low	6.3.52.137
0x0089	NCSI_RXD1 Address High	6.3.52.138
0x008A	NCSI_RXD1 Data Low	6.3.52.139
0x008B	NCSI_RXD1 Data High	6.3.52.140
0x008C	NCSI_CRS_DV Address Low	6.3.52.141
0x008D	NCSI_CRS_DV Address High	6.3.52.142
0x008E	NCSI_CRS_DV Data Low	6.3.52.143
0x008F	NCSI_CRS_DV Data High	6.3.52.144
0x0090	NCSI_ARB_IN Address Low	6.3.52.145
0x0091	NCSI_ARB_IN Address High	6.3.52.146

Table 6-58. Sideband Bus Auto-Load Section Summary Table [continued]

Word Offset	Description	Section Reference
0x0092	NCSI_ARB_IN Data Low	6.3.52.147
0x0093	NCSI_ARB_IN Data High	6.3.52.148
0x0094	NCSI_TX_EN Address Low	6.3.52.149
0x0095	NCSI_TX_EN Address High	6.3.52.150
0x0096	NCSI_TX_EN Data Low	6.3.52.151
0x0097	NCSI_TX_EN Data High	6.3.52.152
0x0098	NCSI_TXD0 Address Low	6.3.52.153
0x0099	NCSI_TXD0 Address High	6.3.52.154
0x009A	NCSI_TXD0 Data Low	6.3.52.155
0x009B	NCSI_TXD0 Data High	6.3.52.156
0x009C	NCSI_TXD1 Address Low	6.3.52.157
0x009D	NCSI_TXD1 Address High	6.3.52.158
0x009E	NCSI_TXD1 Data Low	6.3.52.159
0x009F	NCSI_TXD1 Data High	6.3.52.160
0x00A0	NCSI_ARB_OUT Address Low	6.3.52.161
0x00A1	NCSI_ARB_OUT Address High	6.3.52.162
0x00A2	NCSI_ARB_OUT Data Low	6.3.52.163
0x00A3	NCSI_ARB_OUT Data High	6.3.52.164
0x00A4	CPU_GP_1 Address Low	6.3.52.165
0x00A5	CPU_GP_1 Address High	6.3.52.166
0x00A6	CPU_GP_1 Data Low	6.3.52.167
0x00A7	CPU_GP_1 Data High	6.3.52.168
0x00A8	NAC_GBE_GPIO0 Address Low	6.3.52.169
0x00A9	NAC_GBE_GPIO0 Address High	6.3.52.170
0x00AA	NAC_GBE_GPIO0 Data Low	6.3.52.171
0x00AB	NAC_GBE_GPIO0 Data High	6.3.52.172
0x00AC	NAC_GBE_GPIO1 Address Low	6.3.52.173
0x00AD	NAC_GBE_GPIO1 Address High	6.3.52.174
0x00AE	NAC_GBE_GPIO1 Data Low	6.3.52.175
0x00AF	NAC_GBE_GPIO1 Data High	6.3.52.176
0x00B0	NAC_GBE_GPIO2 Address Low	6.3.52.177
0x00B1	NAC_GBE_GPIO2 Address High	6.3.52.178
0x00B2	NAC_GBE_GPIO2 Data Low	6.3.52.179
0x00B3	NAC_GBE_GPIO2 Data High	6.3.52.180
0x00B4	NAC_GBE_GPIO3 Address Low	6.3.52.181
0x00B5	NAC_GBE_GPIO3 Address High	6.3.52.182
0x00B6	NAC_GBE_GPIO3 Data Low	6.3.52.183

Table 6-58. Sideband Bus Auto-Load Section Summary Table [continued]

Word Offset	Description	Section Reference
0x00B7	NAC_GBE_GPIO3 Data High	6.3.52.184
0x00B8	CDF CFIO East - Type F Word 0	6.3.52.185
0x00B9	CDF CFIO East - Type F Word 1	6.3.52.186
0x00BA	CDF CFIO East - Type F Word 2	6.3.52.187
0x00BB	GPIO0 Address Low	6.3.52.188
0x00BC	GPIO0 Address High	6.3.52.189
0x00BD	GPIO0 Data Low	6.3.52.190
0x00BE	GPIO0 Data High	6.3.52.191
0x00BF	GPIO1 Address Low	6.3.52.192
0x00C0	GPIO1 Address High	6.3.52.193
0x00C1	GPIO1 Data Low	6.3.52.194
0x00C2	GPIO1 Data High	6.3.52.195
0x00C3	GPIO2 Address Low	6.3.52.196
0x00C4	GPIO2 Address High	6.3.52.197
0x00C5	GPIO2 Data Low	6.3.52.198
0x00C6	GPIO2 Data High	6.3.52.199
0x00C7	GPIO3 Address Low	6.3.52.200
0x00C8	GPIO3 Address High	6.3.52.201
0x00C9	GPIO3 Data Low	6.3.52.202
0x00CA	GPIO3 Data High	6.3.52.203
0x00CB	GPIO4 Address Low	6.3.52.204
0x00CC	GPIO4 Address High	6.3.52.205
0x00CD	GPIO4 Data Low	6.3.52.206
0x00CE	GPIO4 Data High	6.3.52.207
0x00CF	GPIO5 Address Low	6.3.52.208
0x00D0	GPIO5 Address High	6.3.52.209
0x00D1	GPIO5 Data Low	6.3.52.210
0x00D2	GPIO5 Data High	6.3.52.211
0x00D3	GPIO6 Address Low	6.3.52.212
0x00D4	GPIO6 Address High	6.3.52.213
0x00D5	GPIO6 Data Low	6.3.52.214
0x00D6	GPIO6 Data High	6.3.52.215
0x00D7	GPIO7 Address Low	6.3.52.216
0x00D8	GPIO7 Address High	6.3.52.217
0x00D9	GPIO7 Data Low	6.3.52.218
0x00DA	GPIO7 Data High	6.3.52.219
0x00DB	GPIO8 Address Low	6.3.52.220

Table 6-58. Sideband Bus Auto-Load Section Summary Table [continued]

Word Offset	Description	Section Reference
0x00DC	GPIO8 Address High	6.3.52.221
0x00DD	GPIO8 Data Low	6.3.52.222
0x00DE	GPIO8 Data High	6.3.52.223
0x00DF	GPIO9 Address Low	6.3.52.224
0x00E0	GPIO9 Address High	6.3.52.225
0x00E1	GPIO9 Data Low	6.3.52.226
0x00E2	GPIO9 Data High	6.3.52.227
0x00E3	GPIO10 Address Low	6.3.52.228
0x00E4	GPIO10 Address High	6.3.52.229
0x00E5	GPIO10 Data Low	6.3.52.230
0x00E6	GPIO10 Data High	6.3.52.231
0x00E7	GPIO11 Address Low	6.3.52.232
0x00E8	GPIO11 Address High	6.3.52.233
0x00E9	GPIO11 Data Low	6.3.52.234
0x00EA	GPIO11 Data High	6.3.52.235
0x00EB	GPIO12 Address Low	6.3.52.236
0x00EC	GPIO12 Address High	6.3.52.237
0x00ED	GPIO12 Data Low	6.3.52.238
0x00EE	GPIO12 Data High	6.3.52.239
0x00EF	Manual Additions	6.3.52.240

6.3.52.1 Module Length (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 1 First Section -> Word: Sideband Bus Auto-Load -> Module Length Last Section -> Word: Sideband Bus Auto-Load -> Manual Additions

6.3.52.2 CDF CFIO West - Type F Word 0 (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Subtype	0x0	
3:0	Type	0xF	

6.3.52.3 CDF CFIO West - Type F Word 1 (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15	R	0b	
14:12	TAG	000b	
11	AL	0b	
10:8	BAR	000b	
7:4	Select ID	0x7	
3:0	Opcode	0x7	

6.3.52.4 CDF CFIO West - Type F Word 2 (0x0003)

Bit(s)	Field Name	Default NVM Value	Description
15:12	Second Byte En	0x0	
11:8	First Byte En	0x2	
7:0	Function ID	0x0	

6.3.52.5 GBE_SDP_TIMESYNC0 Address Low (0x0004)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x800	
3:0	Type	0x1	

6.3.52.6 GBE_SDP_TIMESYNC0 Address High (0x0005)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.7 GBE_SDP_TIMESYNC0 Data Low (0x0006)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.8 GBE_SDP_TIMESYNC0 Data High (0x0007)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.9 GBE_SDP_TIMESYNC1 Address Low (0x0008)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x810	
3:0	Type	0x1	

6.3.52.10 GBE_SDP_TIMESYNC1 Address High (0x0009)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.11 GBE_SDP_TIMESYNC1 Data Low (0x000A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.12 GBE_SDP_TIMESYNC1 Data High (0x000B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.13 GBE_SDP_TIMESYNC2 Address Low (0x000C)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x820	
3:0	Type	0x1	

6.3.52.14 GBE_SDP_TIMESYNC2 Address High (0x000D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.15 GBE_SDP_TIMESYNC2 Data Low (0x000E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.16 GBE_SDP_TIMESYNC2 Data High (0x000F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.17 GBE_SDP_TIMESYNC3 Address Low (0x0010)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x830	
3:0	Type	0x1	

6.3.52.18 GBE_SDP_TIMESYNC3 Address High (0x0011)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.19 GBE_SDP_TIMESYNC3 Data Low (0x0012)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.20 GBE_SDP_TIMESYNC3 Data High (0x0013)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.21 GBE0_I2C_CLK Address Low (0x0014)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x840	
3:0	Type	0x1	

6.3.52.22 GBE0_I2C_CLK Address High (0x0015)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.23 GBE0_I2C_CLK Data Low (0x0016)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.24 GBE0_I2C_CLK Data High (0x0017)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.25 GBE0_I2C_DATA Address Low (0x0018)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x850	
3:0	Type	0x1	

6.3.52.26 GBE0_I2C_DATA Address High (0x0019)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.27 GBE0_I2C_DATA Data Low (0x001A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.28 GBE0_I2C_DATA Data High (0x001B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.29 GBE1_I2C_CLK Address Low (0x001C)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x860	
3:0	Type	0x1	

6.3.52.30 GBE1_I2C_CLK Address High (0x001D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.31 GBE1_I2C_CLK Data Low (0x001E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.32 GBE1_I2C_CLK Data High (0x001F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.33 GBE1_I2C_DATA Address Low (0x0020)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x870	
3:0	Type	0x1	

6.3.52.34 GBE1_I2C_DATA Address High (0x0021)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.35 GBE1_I2C_DATA Data Low (0x0022)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.36 GBE1_I2C_DATA Data High (0x0023)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.37 GBE2_I2C_CLK Address Low (0x0024)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x880	
3:0	Type	0x1	

6.3.52.38 GBE2_I2C_CLK Address High (0x0025)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.39 GBE2_I2C_CLK Data Low (0x0026)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.40 GBE2_I2C_CLK Data High (0x0027)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.41 GBE2_I2C_DATA Address Low (0x0028)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x890	
3:0	Type	0x1	

6.3.52.42 GBE2_I2C_DATA Address High (0x0029)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.43 GBE2_I2C_DATA Data Low (0x002A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.44 GBE2_I2C_DATA Data High (0x002B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.45 GBE3_I2C_CLK Address Low (0x002C)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x8A0	
3:0	Type	0x1	

6.3.52.46 GBE3_I2C_CLK Address High (0x002D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.47 GBE3_I2C_CLK Data Low (0x002E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.48 GBE3_I2C_CLK Data High (0x002F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.49 GBE3_I2C_DATA Address Low (0x0030)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x8B0	
3:0	Type	0x1	

6.3.52.50 GBE3_I2C_DATA Address High (0x0031)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.51 GBE3_I2C_DATA Data Low (0x0032)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.52 GBE3_I2C_DATA Data High (0x0033)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.53 GBE0_LED0 Address Low (0x0034)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x0C8	
3:0	Type	0x1	

6.3.52.54 GBE0_LED0 Address High (0x0035)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.55 GBE0_LED0 Data Low (0x0036)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.56 GBE0_LED0 Data High (0x0037)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.57 GBE0_LED1 Address Low (0x0038)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x8D0	
3:0	Type	0x1	

6.3.52.58 GBE0_LED1 Address High (0x0039)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.59 GBE0_LED1 Data Low (0x003A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.60 GBE0_LED1 Data High (0x003B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.61 GBE0_LED2 Address Low (0x003C)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x8E0	
3:0	Type	0x1	

6.3.52.62 GBE0_LED2 Address High (0x003D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.63 GBE0_LED2 Data Low (0x003E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.64 GBE0_LED2 Data High (0x003F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.65 GBE1_LED0 Address Low (0x0040)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x8F0	
3:0	Type	0x1	

6.3.52.66 GBE1_LED0 Address High (0x0041)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.67 GBE1_LED0 Data Low (0x0042)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.68 GBE1_LED0 Data High (0x0043)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.69 GBE1_LED1 Address Low (0x0044)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x900	
3:0	Type	0x1	

6.3.52.70 GBE1_LED1 Address High (0x0045)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.71 GBE1_LED1 Data Low (0x0046)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.72 GBE1_LED1 Data High (0x0047)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.73 GBE1_LED2 Address Low (0x0048)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x910	
3:0	Type	0x1	

6.3.52.74 GBE1_LED2 Address High (0x0049)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.75 GBE1_LED2 Data Low (0x004A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.76 GBE1_LED2 Data High (0x004B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.77 GBE2_LED0 Address Low (0x004C)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x920	
3:0	Type	0x1	

6.3.52.78 GBE2_LED0 Address High (0x004D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.79 GBE2_LED0 Data Low (0x004E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.80 GBE2_LED0 Data High (0x004F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.81 GBE2_LED1 Address Low (0x0050)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x930	
3:0	Type	0x1	

6.3.52.82 GBE2_LED1 Address High (0x0051)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.83 GBE2_LED1 Data Low (0x0052)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.84 GBE2_LED1 Data High (0x0053)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.85 GBE2_LED2 Address Low (0x0054)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x940	
3:0	Type	0x1	

6.3.52.86 GBE2_LED2 Address High (0x0055)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.87 GBE2_LED2 Data Low (0x0056)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.88 GBE2_LED2 Data High (0x0057)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.89 GBE3_LED0 Address Low (0x0058)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x950	
3:0	Type	0x1	

6.3.52.90 GBE3_LED0 Address High (0x0059)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.91 GBE3_LED0 Data Low (0x005A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.92 GBE3_LED0 Data High (0x005B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.93 GBE3_LED1 Address Low (0x005C)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x960	
3:0	Type	0x1	

6.3.52.94 GBE3_LED1 Address High (0x005D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.95 GBE3_LED1 Data Low (0x005E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.96 GBE3_LED1 Data High (0x005F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.97 GBE3_LED2 Address Low (0x0060)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x970	
3:0	Type	0x1	

6.3.52.98 GBE3_LED2 Address High (0x0061)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.99 GBE3_LED2 Data Low (0x0062)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.100 GBE3_LED2 Data High (0x0063)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.101 GBE_SMB_CLK Address Low (0x0064)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0xA10	
3:0	Type	0x1	

6.3.52.102 GBE_SMB_CLK Address High (0x0065)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.103 GBE_SMB_CLK Data Low (0x0066)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.104 GBE_SMB_CLK Data High (0x0067)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.105 GBE_SMB_DATA Address Low (0x0068)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0xA20	
3:0	Type	0x1	

6.3.52.106 GBE_SMB_DATA Address High (0x0069)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.107 GBE_SMB_DATA Data Low (0x006A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.108 GBE_SMB_DATA Data High (0x006B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.109 GBE_SMB_ALRT_N Address Low (0x006C)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0xA30	
3:0	Type	0x1	

6.3.52.110 GBE_SMB_ALRT_N Address High (0x006D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.111 GBE_SMB_ALRT_N Data Low (0x006E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.112 GBE_SMB_ALRT_N Data High (0x006F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.113 UART1_RXD Address Low (0x0070)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0xAE0	
3:0	Type	0x1	

6.3.52.114 UART1_RXD Address High (0x0071)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.115 UART1_RXD Data Low (0x0072)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.116 UART1_RXD Data High (0x0073)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.117 UART1_TXD Address Low (0x0074)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0xAF0	
3:0	Type	0x1	

6.3.52.118 UART1_TXD Address High (0x0075)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.119 UART1_TXD Data Low (0x0076)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.120 UART1_TXD Data High (0x0077)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.121 CPU_GP_0 Address Low (0x0078)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0xB00	
3:0	Type	0x1	

6.3.52.122 CPU_GP_0 Address High (0x0079)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.123 CPU_GP_0 Data Low (0x007A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.124 CPU_GP_0 Data High (0x007B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.125 GBE_MNG_I2C_CLK Address Low (0x007C)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0xC50	
3:0	Type	0x1	

6.3.52.126 GBE_MNG_I2C_CLK Address High (0x007D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.127 GBE_MNG_I2C_CLK Data Low (0x007E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.128 GBE_MNG_I2C_CLK Data High (0x007F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.129 GBE_MNG_I2C_DATA Address Low (0x0080)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0xC60	
3:0	Type	0x1	

6.3.52.130 GBE_MNG_I2C_DATA Address High (0x0081)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.131 GBE_MNG_I2C_DATA Data Low (0x0082)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.132 GBE_MNG_I2C_DATA Data High (0x0083)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.133 NCSI_RXD0 Address Low (0x0084)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x980	
3:0	Type	0x1	

6.3.52.134 NCSI_RXD0 Address High (0x0085)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.135 NCSI_RXD0 Data Low (0x0086)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.136 NCSI_RXD0 Data High (0x0087)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.137 NCSI_RXD1 Address Low (0x0088)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x9A0	
3:0	Type	0x1	

6.3.52.138 NCSI_RXD1 Address High (0x0089)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.139 NCSI_RXD1 Data Low (0x008A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.140 NCSI_RXD1 Data High (0x008B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.141 NCSI_CRS_DV Address Low (0x008C)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x9B0	
3:0	Type	0x1	

6.3.52.142 NCSI_CRS_DV Address High (0x008D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.143 NCSI_CRS_DV Data Low (0x008E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.144 NCSI_CRS_DV Data High (0x008F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.145 NCSI_ARB_IN Address Low (0x0090)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x9C0	
3:0	Type	0x1	

6.3.52.146 NCSI_ARB_IN Address High (0x0091)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.147 NCSI_ARB_IN Data Low (0x0092)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.148 NCSI_ARB_IN Data High (0x0093)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.149 NCSI_TX_EN Address Low (0x0094)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x9D0	
3:0	Type	0x1	

6.3.52.150 NCSI_TX_EN Address High (0x0095)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.151 NCSI_TX_EN Data Low (0x0096)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.152 NCSI_TX_EN Data High (0x0097)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.153 NCSI_TXD0 Address Low (0x0098)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x9E0	
3:0	Type	0x1	

6.3.52.154 NCSI_TXD0 Address High (0x0099)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.155 NCSI_TXD0 Data Low (0x009A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.156 NCSI_TXD0 Data High (0x009B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.157 NCSI_TXD1 Address Low (0x009C)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x9F0	
3:0	Type	0x1	

6.3.52.158 NCSI_TXD1 Address High (0x009D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.159 NCSI_TXD1 Data Low (0x009E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.160 NCSI_TXD1 Data High (0x009F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.161 NCSI_ARB_OUT Address Low (0x00A0)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0xA00	
3:0	Type	0x1	

6.3.52.162 NCSI_ARB_OUT Address High (0x00A1)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.163 NCSI_ARB_OUT Data Low (0x00A2)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.164 NCSI_ARB_OUT Data High (0x00A3)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.165 CPU_GP_1 Address Low (0x00A4)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0xB10	
3:0	Type	0x1	

6.3.52.166 CPU_GP_1 Address High (0x00A5)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.167 CPU_GP_1 Data Low (0x00A6)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.168 CPU_GP_1 Data High (0x00A7)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.169 NAC_GBE_GPIO0 Address Low (0x00A8)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x170	
3:0	Type	0x1	

6.3.52.170 NAC_GBE_GPIO0 Address High (0x00A9)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0001	

6.3.52.171 NAC_GBE_GPIO0 Data Low (0x00AA)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.172 NAC_GBE_GPIO0 Data High (0x00AB)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.173 NAC_GBE_GPIO1 Address Low (0x00AC)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x180	
3:0	Type	0x1	

6.3.52.174 NAC_GBE_GPIO1 Address High (0x00AD)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0001	

6.3.52.175 NAC_GBE_GPIO1 Data Low (0x00AE)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.176 NAC_GBE_GPIO1 Data High (0x00AF)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.177 NAC_GBE_GPIO2 Address Low (0x00B0)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x190	
3:0	Type	0x1	

6.3.52.178 NAC_GBE_GPIO2 Address High (0x00B1)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0001	

6.3.52.179 NAC_GBE_GPIO2 Data Low (0x00B2)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.180 NAC_GBE_GPIO2 Data High (0x00B3)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.181 NAC_GBE_GPIO3 Address Low (0x00B4)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x1A0	
3:0	Type	0x1	

6.3.52.182 NAC_GBE_GPIO3 Address High (0x00B5)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0001	

6.3.52.183 NAC_GBE_GPIO3 Data Low (0x00B6)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.184 NAC_GBE_GPIO3 Data High (0x00B7)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.185 CDF CFIO East - Type F Word 0 - 0x00B8

Bit(s)	Field Name	Default NVM Value	Description
15:4	Subtype	0x0	
3:0	Type	0xF	

6.3.52.186 CDF CFIO East - Type F Word 1 - 0x00B9

Bit(s)	Field Name	Default NVM Value	Description
15	R	0b	
14:12	TAG	000b	
11	AL	0b	
10:8	BAR	000b	
7:4	Select ID	0x8	
3:0	Opcode	0x7	

6.3.52.187 CDF CFIO East - Type F Word 2 - 0x00BA

Bit(s)	Field Name	Default NVM Value	Description
15:12	Second Byte En	0x0	
11:8	First Byte En	0x2	
7:0	Function ID	0x0	

6.3.52.188 GPIO0 Address Low (0x00BB)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x810	
3:0	Type	0x1	

6.3.52.189 GPIO0 Address High (0x00BC)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.190 GPIO0 Data Low (0x00BD)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.191 GPIO0 Data High (0x00BE)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.192 GPIO1 Address Low (0x00BF)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x820	
3:0	Type	0x1	

6.3.52.193 GPIO1 Address High (0x00C0)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.194 GPIO1 Data Low (0x00C1)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.195 GPIO1 Data High (0x00C2)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.196 GPIO2 Address Low (0x00C3)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x830	
3:0	Type	0x1	

6.3.52.197 GPIO2 Address High (0x00C4)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.198 GPIO2 Data Low (0x00C5)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.199 GPIO2 Data High (0x00C6)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.200 GPIO3 Address Low (0x00C7)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x840	
3:0	Type	0x1	

6.3.52.201 GPIO3 Address High (0x00C8)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.202 GPIO3 Data Low (0x00C9)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.203 GPIO3 Data High (0x00CA)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.204 GPIO4 Address Low (0x00CB)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x850	
3:0	Type	0x1	

6.3.52.205 GPIO4 Address High (0x00CC)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.206 GPIO4 Data Low (0x00CD)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.207 GPIO4 Data High (0x00CE)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.208 GPIO5 Address Low (0x00CF)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x860	
3:0	Type	0x1	

6.3.52.209 GPIO5 Address High (0x00D0)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.210 GPIO5 Data Low (0x00D1)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.211 GPIO5 Data High (0x00D2)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.212 GPIO6 Address Low (0x00D3)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x870	
3:0	Type	0x1	

6.3.52.213 GPIO6 Address High (0x00D4)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.214 GPIO6 Data Low (0x00D5)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.215 GPIO6 Data High (0x00D6)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.216 GPIO7 Address Low (0x00D7)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x880	
3:0	Type	0x1	

6.3.52.217 GPIO7 Address High (0x00D8)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.218 GPIO7 Data Low (0x00D9)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.219 GPIO7 Data High (0x00DA)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.220 GPIO8 Address Low (0x00DB)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x890	
3:0	Type	0x1	

6.3.52.221 GPIO8 Address High (0x00DC)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.222 GPIO8 Data Low (0x00DD)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.223 GPIO8 Data High (0x00DE)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.224 GPIO9 Address Low (0x00DF)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x8A0	
3:0	Type	0x1	

6.3.52.225 GPIO9 Address High (0x00E0)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.226 GPIO9 Data Low (0x00E1)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.227 GPIO9 Data High (0x00E2)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.228 GPIO10 Address Low (0x00E3)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x8B0	
3:0	Type	0x1	

6.3.52.229 GPIO10 Address High (0x00E4)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.230 GPIO10 Data Low (0x00E5)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.231 GPIO10 Data High (0x00E6)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.232 GPIO11 Address Low (0x00E7)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x8C0	
3:0	Type	0x1	

6.3.52.233 GPIO11 Address High (0x00E8)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.234 GPIO11 Data Low (0x00E9)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.235 GPIO11 Data High (0x00EA)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.236 GPIO12 Address Low (0x00EB)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Address Low	0x8D0	
3:0	Type	0x1	

6.3.52.237 GPIO12 Address High (0x00EC)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Address High	0x0	

6.3.52.238 GPIO12 Data Low (0x00ED)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data Low	0x0700	Valid values are: 0x0700 = Function 1 0x0B00 = Function 2 0x0F00 = Function 3 0x1300 = Function 4

6.3.52.239 GPIO12 Data High (0x00EE)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Data High	0x0	

6.3.52.240 Manual Additions (0x00EF)

Raw data module length: variable

6.3.53 EMP SR Settings Module Header Section

This section contains the modes of operation of the EMP which are be stored in the Shadow RAM.

Table 6-59. EMP SR Settings Module Header Section Summary Table

Word Offset	Description	Section Reference
0x0000	Section Header	6.3.53.1
0x0001	SMBus Slave Addresses	6.3.53.2
0x0002	SMBus Slave Addresses 2	6.3.53.3
0x0003	SMBus Slave Addresses 3	6.3.53.4
0x0004	SMBus Slave Addresses 4	6.3.53.5
0x0005	OEM Capabilities	6.3.53.6
0x0006	OEM Technologies Enabled	6.3.53.7
0x0007	SR PF Allocations Pointer	6.3.53.8
0x0008	Max PF and VF per Port	6.3.53.9
0x0009	PF LAN Device ID	6.3.53.10
0x000A	PF SAN Device ID	6.3.53.11
0x000B	PF iSCSI Device ID	6.3.53.12
0x000C	PF RDMA Device ID	6.3.53.13
0x000D	VF LAN Device ID	6.3.53.14
0x000E	VF SAN Device ID	6.3.53.15
0x000F	VF iSCSI Device ID	6.3.53.16
0x0010	VF RDMA Device ID	6.3.53.17
0x0011	PF LAN Subsystem ID	6.3.53.18
0x0012	PF SAN Subsystem ID	6.3.53.19
0x0013	PF iSCSI Subsystem ID	6.3.53.20
0x0014	PF RDMA Subsystem ID	6.3.53.21
0x0015	VF LAN Subsystem ID	6.3.53.22
0x0016	VF SAN Subsystem ID	6.3.53.23
0x0017	VF iSCSI Subsystem ID	6.3.53.24
0x0018	VF RDMA Subsystem ID	6.3.53.25
0x0019 + 1*n, n=0...7	PFGEN_STATE	6.3.53.26
0x0021	Reserved	6.3.53.27
0x0022	Features Enable	6.3.53.28
0x0023	LLDP Configuration Pointer	6.3.53.29
0x0024	Allowed SB Targets	6.3.53.30
0x0025	Allow 64 Bits Transactions	6.3.53.31
0x0026	Allowed Opcodes	6.3.53.32

6.3.53.1 Section Header (0x0000)

Bit(s)	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 -> Allowed Opcodes Section length in words.

6.3.53.2 SMBus Slave Addresses (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:9	SMBus 1 Slave Address	0x4A	Dual address mode only.
8	Reserved	0b	Reserved.
7:1	SMBus 0 Slave Address	0x49	
0	Reserved	0b	Reserved.

6.3.53.3 SMBus Slave Addresses 2 (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:9	SMBus 3 Slave Address	0x4C	
8	Reserved	0b	Reserved.
7:1	SMBus 2 Slave Address	0x4B	
0	Reserved	0b	Reserved.

6.3.53.4 SMBus Slave Addresses 3 (0x0003)

Bit(s)	Field Name	Default NVM Value	Description
15:9	SMBus 5 Slave Address	0x4E	Dual address mode only.
8	Reserved	0b	Reserved.
7:1	SMBus 4 Slave Address	0x4D	
0	Reserved	0b	Reserved.

6.3.53.5 SMBus Slave Addresses 4 (0x0004)

Bit(s)	Field Name	Default NVM Value	Description
15:9	SMBus 7 Slave Address	0x50	
8	Reserved	0b	Reserved.
7:1	SMBus 6 Slave Address	0x4F	
0	Reserved	0b	Reserved.

6.3.53.6 OEM Capabilities (0x0005)

Defines the OEM technologies supported in this device.

Bit(s)	Field Name	Default NVM Value	Description
15:7	Reserved	0x0	Reserved.
6	OCBB over MCTP Capable	0b	HP OCBB over MCTP Capable enable. 0b = Disabled 1b = Capable
5	HP PLDM support Capable	0b	HP PLDM support Capable. 0b = Disabled 1b = Capable
4:0	Reserved	0x0	Reserved.

6.3.53.7 OEM Technologies Enabled (0x0006)

Defines the OEM technologies enabled in this device.

Bit(s)	Field Name	Default NVM Value	Description
15:14	Reserved	00b	Reserved.
13	Anti Rollback Feature Enable	0b	Anti Rollback feature enable. 0b = Disabled 1b = Enabled
12	OCBB over MCTP	0b	HP OCBB enable. 0b = Disabled 1b = Enabled
11	HP PLDM Support	0b	Enable PLDM support. 0b = Disabled 1b = Enabled
10:0	Reserved	0x0	Reserved.

6.3.53.8 SR PF Allocations Pointer (0x0007)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PF Allocations Pointer	0xFFFF	Points to SR PF Allocations Section. For SR PF Allocations inner structure, see Section 6.3.54.

6.3.53.9 Max PF and VF per Port (0x0008)

Bit(s)	Field Name	Default NVM Value	Description
15:8	Max VF per PF	0x80	Max number of VFs allocated to a PF.
7:3	Reserved	0x0	Reserved.
2:0	Max PF per Port	010b	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

6.3.53.10 PF LAN Device ID (0x0009)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PF LAN Device ID	0x1888	

6.3.53.11 PF SAN Device ID (0x000A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PF SAN Device ID	0x1888	

6.3.53.12 PF iSCSI Device ID (0x000B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PF iSCSI Device ID	0x1888	

6.3.53.13 PF RDMA Device ID (0x000C)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PF RDMA Device ID	0x1888	

6.3.53.14 VF LAN Device ID (0x000D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	VF LAN Device ID	0x1889	

6.3.53.15 VF SAN Device ID (0x000E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	VF SAN Device ID	0x1889	

6.3.53.16 VF iSCSI Device ID (0x000F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	VF iSCSI Device ID	0x1889	

6.3.53.17 VF RDMA Device ID (0x0010)

Bit(s)	Field Name	Default NVM Value	Description
15:0	VF RDMA Device ID	0x1889	

6.3.53.18 PF LAN Subsystem ID (0x0011)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PF LAN Subsystem ID	0x0	

6.3.53.19 PF SAN Subsystem ID (0x0012)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PF SAN Subsystem ID	0x0	

6.3.53.20 PF iSCSI Subsystem ID (0x0013)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PF iSCSI Subsystem ID	0x0	

6.3.53.21 PF RDMA Subsystem ID (0x0014)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PF RDMA Subsystem ID	0x0	

6.3.53.22 VF LAN Subsystem ID (0x0015)

Bit(s)	Field Name	Default NVM Value	Description
15:0	VF LAN Subsystem ID	0x0	

6.3.53.23 VF SAN Subsystem ID (0x0016)

Bit(s)	Field Name	Default NVM Value	Description
15:0	VF SAN Subsystem ID	0x0	

6.3.53.24 VF iSCSI Subsystem ID (0x0017)

Bit(s)	Field Name	Default NVM Value	Description
15:0	VF iSCSI Subsystem ID	0x0	

6.3.53.25 VF RDMA Subsystem ID (0x0018)

Bit(s)	Field Name	Default NVM Value	Description
15:0	VF RDMA Subsystem ID	0x0	

6.3.53.26 PFGEN_STATE[n] (0x0019 + 1*n, n=0...7)

Bit(s)	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	Reserved.
11	PFGEN_STATE[1 + 2*n].PFSCEN	0b	
10	PFGEN_STATE[1 + 2*n].PFLINKEN	1b	
9	PFGEN_STATE[1 + 2*n].PFFCEN	1b	
8	PFGEN_STATE[1 + 2*n].PFPEEN	0b	
7:4	Reserved	0x0	Reserved.
3	PFGEN_STATE[0 + 2*n].PFSCEN	0b	
2	PFGEN_STATE[0 + 2*n].PFLINKEN	1b	
1	PFGEN_STATE[0 + 2*n].PFFCEN	1b	
0	PFGEN_STATE[0 + 2*n].PFPEEN	0b	

6.3.53.27 Reserved (0x0021)

6.3.53.28 Features Enable (0x0022)

Describes support for various features.

Bit(s)	Field Name	Default NVM Value	Description
15	PHY 2 Autoload Enable	1b	PHY Auto-load Enable.
14	PHY 1 Autoload Enable	1b	PHY Auto-load Enable.
13	PHY 0 Autoload Enable	1b	PHY Auto-load Enable.
12	PXE Mode No-Drop Policy Supported	0b	The value indicates if the PXE Mode No-Drop policy is supported. 0b = No support. 1b = 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.
11:1	Reserved	0x0	Reserved.
0	Non-BTS Switch mode	0b	non-BTS mode. ***Note:*** This field is preserved by Intel NVM Update Tool.

6.3.53.29 LLDP Configuration Pointer (0x0023)

Bit(s)	Field Name	Default NVM Value	Description
15:0	LLDP Configuration Pointer	0xFFFF	Points to LLDP Configuration Section. For LLDP Configuration inner structure, see Section 6.3.55.

6.3.53.30 Allowed SB Targets (0x0024)

The list of allowed sideband targets.

Bits	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	Reserved.
11	SPI	0b	
10	FC	0b	
9	DSA	0b	
8	LCB	1b	
7	PCIW	1b	
6	CAR	1b	
5	IO Widget	1b	
4:3	Reserved	00b	Reserved.
2	ETHW	1b	
1:0	Reserved	0b	Reserved.

6.3.53.31 Allow 64 Bits Transactions (0x0025)

Block/Allow 64 bits transactions.

Bits	Field Name	Default NVM Value	Description
15:1	Reserved		Reserved.
0	Allow 64 Bits Transactions	0b	

6.3.53.32 Allowed Opcodes (0x0026)

The list of allowed sideband opcodes.

Bits	Field Name	Default NVM Value	Description
15	Reserved	0b	Reserved.
14	IOWr - IO Register Write (Non Posted)	0b	
13	IOWr - IO Register Write (Posted)	0b	
12	IORd - IO Register Read (Non Posted)	0b	
11	CfgWr - Config Register Write (Non Posted)	1b	
10	CfgWr - Config Register Write (Posted)	0b	
9	CfgRd - Config Register Read (Non Posted)	1b	
8	CRRd - Read Private Control Register (Non Posted)	0b	
7	CRWr - Write Private Control Register (Non Posted)	0b	
6	CRWr - Write Private Control Register (Posted)	0b	
5	MBWr - Multiple Block Write (Non Posted)	0b	
4	MBWr - Multiple Block Write (Posted)	0b	
3	MBRd - Multiple Block Read (Non Posted)	0b	
2	MWr - Register Write (Non Posted)	0b	
1	MWr - Register Write (Posted)	1b	
0	Mrd - Register Read (Non Posted)	1b	

6.3.54 SR PF Allocations Section

This section contains the modes of operation of the EMP which are be stored in the Shadow RAM.

Table 6-60. SR PF Allocations Section Summary Table

Word Offset	Description	Section Reference
0x0000	Header	6.3.54.1
0x0001 + 2*n, n=0...7	PF Flags	6.3.54.2
0x0002 + 2*n, n=0...7	PF BW	6.3.54.3
0x0011 + 2*n, n=0...23	PF Allocations - Type	6.3.54.4
0x0012 + 2*n, n=0...23	PF Allocations - Value	6.3.54.5

6.3.54.1 Header (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Block Length		Length in: 2 Bytes unit - 1 First Section -> Word: SR PF Allocations -> Header Last Section -> Word: SR PF Allocations -> PF Allocations - Value Size of section.

6.3.54.2 PF Flags[n] (0x0001 + 2*n, n=0...7)

Bit(s)	Field Name	Default NVM Value	Description
15:1	Reserved	0x0	Reserved.
0	Load PF MAC Address	1b	Defines if the LAN MAC Address of the PF should be added to the filtering table. If this bit is set, only packets that passes the MAC and if needed, the STag are forwarded to the PF.

6.3.54.3 PF BW[n] (0x0002 + 2*n, n=0...7)

Bit(s)	Field Name	Default NVM Value	Description
15:8	PF Max BW	0x64	This field contains the Maximum Tx bandwidth allocation of the specified partition expressed in % of the Maximum physical port link speed. The % value ranges from 0 to 100.
7:0	PF Min BW	0x0	This field contains the Minimum Tx bandwidth allocation of the specified partition expressed in % of the Maximum physical port link speed. The % value ranges from 0 to 100.

6.3.54.4 PF Allocations - Type[n] (0x0011 + 2*n, n=0...23)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Type	0xFFFF	Type array.

6.3.54.5 PF Allocations - Value[n] (0x0012 + 2*n, n=0...23)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Value	0x0	Value array.

6.3.55 LLDP Configuration Section

Default settings to the embedded LLDP agent.

Table 6-61. LLDP Configuration Section Summary Table

Word Offset	Description	Section Reference
0x0000	Section Length	6.3.55.1
0x0001	LLDP Admin Status 0	6.3.55.2
0x0002	LLDP Admin Status 1	6.3.55.3
0x0003	msgFastTx	6.3.55.4
0x0004	msgTxInterval	6.3.55.5
0x0005	LLDP Tx Parameters	6.3.55.6
0x0006	LLDP Initialization Timers	6.3.55.7
0x0007	ENDLESS_XOFF_THRESH	6.3.55.8
0x0008	DCBX Mode 0	6.3.55.9
0x0009	DCBX Mode 1	6.3.55.10

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.

Bit(s)	Field Name	Default NVM Value	Description
15:0	Section Length		Length in: 2 Bytes unit - 1 First Section -> Word: LLDP Configuration -> Section Length Last Section -> Word: LLDP Configuration -> DCBX Mode 1

6.3.55.2 LLDP Admin Status 0 (0x0001)

Defines status of LLDP agent. Each LAN port has independent status.

0: LLDP agent is disabled.
1: LLDP is configured for receives only.
2: LLDP is configured for transmits only.
3: Both receive and transmit enabled (default).

Bit(s)	Field Name	Default NVM Value	Description
15:12	Port 3	0x3	Defines status of LLDP agent. Applies to LAN Port 3.
11:8	Port 2	0x3	Defines status of LLDP agent. Applies to LAN Port 2.
7:4	Port 1	0x3	Defines status of LLDP agent. Applies to LAN Port 1.
3:0	Port 0	0x3	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.55.3 LLDP Admin Status 1 (0x0002)

Defines status of LLDP agent. Each LAN port has independent status.

0: LLDP agent is disabled.
1: LLDP is configured for receives only.
2: LLDP is configured for transmits only.
3: Both receive and transmit enabled (default).

Bit(s)	Field Name	Default NVM Value	Description
15:12	Port 7	0x3	Defines status of LLDP agent. Applies to LAN Port 7.
11:8	Port 6	0x3	Defines status of LLDP agent. Applies to LAN Port 6.
7:4	Port 5	0x3	Defines status of LLDP agent. Applies to LAN Port 5.
3:0	Port 4	0x3	Defines status of LLDP agent. Applies to LAN Port 4.

6.3.55.4 msgFastTx (0x0003)

Bit(s)	Field Name	Default NVM Value	Description
15:0	msgFastTx	0x1	Time interval in timer ticks (Seconds) between PDU transmits during fast transmits period.

6.3.55.5 msgTxInterval (0x0004)

Bit(s)	Field Name	Default NVM Value	Description
15:0	msgTxInterval	0x1E	Is the time in timer ticks (Seconds) between transmissions during normal transmission.

6.3.55.6 LLDP Tx Parameters (0x0005)

Bit(s)	Field Name	Default NVM Value	Description
15:8	txCreditMax	0x5	Determines maximum number of LLDPDUs that can be sent per second.
7:0	msgTxHold	0x4	Is used as a multiplier of *msgTxInterval* to determine the txTTL that is carried in the LLDP frames.

6.3.55.7 LLDP Initialization Timers (0x0006)

Timers used by LLDP agent during initialization and when to reinitialize. All times are in seconds.

Bit(s)	Field Name	Default NVM Value	Description
15:8	reinitDelay	0x2	This parameter indicates the amount of delay, in seconds, from when adminStatus becomes disabled until re-initialization is attempted.
7:0	txFastInit	0x4	This variable is used as the initial value for the txFast variable. This value determines the number of LLDPDUs that are transmitted during a fast transmission period.

6.3.55.8 ENDLESS_XOFF_THRESH (0x0007)

Define a time limit the firmware waits for XOFF condition during PFR flow. The time is defined in 1msec units. A possible default setting is 10msec (DCR175 - PFR XOFF deadlock prevention).

Bit(s)	Field Name	Default NVM Value	Description
15:0	ENDLESS_XOFF_THRESH	0xA	Define a time limit the firmware waits for XOFF condition during PFR flow. The time is defined in 1 ms units. A possible default setting is 10 ms.

6.3.55.9 DCBX Mode 0 (0x0008)

Defines per-PORT DCBX mode. Please note that each port could be configured independently regardless of the other ports configurations.

0: No DCBX is supported.
1: IEEE DCBX only.
2: CEE DCBX only.
3: Auto-Select, starting in IEEE (default).

Bit(s)	Field Name	Default NVM Value	Description
15:12	DCBX Port[3]	0x3	Port[3] DCBX mode.
11:8	DCBX Port[2]	0x3	Port[2] DCBX mode.
7:4	DCBX Port[1]	0x3	Port[1] DCBX mode.
3:0	DCBX Port[0]	0x3	Port[0] DCBX mode.

6.3.55.10 DCBX Mode 1 (0x0009)

Defines per-PORT DCBX mode. Please note that each port could be configured independently regardless of the other ports configurations.

0: No DCBX is supported.
1: IEEE DCBX only.
2: CEE DCBX only.
3: Auto-Select, starting in IEEE (default).

Bit(s)	Field Name	Default NVM Value	Description
15:12	DCBX Port[7]	0x3	Port[7] DCBX mode.
11:8	DCBX Port[6]	0x3	Port[6] DCBX mode.
7:4	DCBX Port[5]	0x3	Port[5] DCBX mode.
3:0	DCBX Port[4]	0x3	Port[4] DCBX mode.

6.3.56 GFID Module Section

Table 6-62. GFID Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	Length	6.3.56.1
0x0001 + 1*n, n=0...17	GFID	6.3.56.2
0x0013 + 1*n, n=0...17	Original GFID	6.3.56.3

6.3.56.1 Length (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 2 First Section -> Word: GFID Module -> Length Last Section -> Word: GFID Module -> Original GFID

6.3.56.2 GFID[n] (0x0001 + 1*n, n=0...17)

Bit(s)	Field Name	Default NVM Value	Description
15:0	GFID	0x0	

6.3.56.3 Original GFID[n] (0x0013 + 1*n, n=0...17)

Bits	Field Name	Default NVM Value	Description
15:0	Original GFID	0x0	

6.3.57 Manageability Module Header Section

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.

Table 6-63. Manageability Module Header Section Summary Table

Word Offset	Description	Section Reference
0x0000	Section Length	6.3.57.1
0x0001	Common Manageability Parameters	6.3.57.2
0x0002	Common Manageability Parameters 2	6.3.57.3
0x0003	PLDM Control Word	6.3.57.4
0x0004	RDE Control Word	6.3.57.5
0x0005	OCP NIC Parameters	6.3.57.6
0x0006	Reserved	6.3.57.7
0x0007	Sideband Configuration Pointer	6.3.57.8
0x0008	Reserved	6.3.57.9
0x0009	Traffic Types Parameters	6.3.57.10
0x000A	OEM Section Pointer	6.3.57.11

6.3.57.1 Section Length (0x0000)

The length of the section in words. Note that section length does not include a count for the section length word.

Bit(s)	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

6.3.57.2 Common Manageability Parameters (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:11	Reserved	0x0	Reserved.
10:8	Manageability Pass-Through Mode	010b	Valid values are: 000b = None 010b = Pass-through (PT) mode
7:5	Reserved	000b	Reserved.
4	Force TCO Reset Disable	1b	If cleared, allows the BMC to do a Global reset of the device using the Force TCO SMBus or NC-SI commands. 0b = Enable Force TCO reset. 1b = Disable Force TCO reset.
3	Reserved	0b	Reserved.
2	OS2BMC Capable	1b	0b = Disable 1b = Enable.

Bit(s)	Field Name	Default NVM Value	Description
1:0	Reserved	0b	Reserved.

6.3.57.3 Common Manageability Parameters 2 (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15	LAN7_FTCO_ISOL_DIS	1b	Allow isolation of port 7 per BMC request. 0b = Enabled 1b = Disabled
14	LAN6_FTCO_ISOL_DIS	1b	Allow isolation of port 6 per BMC request. 0b = Enabled 1b = Disabled
13	LAN5_FTCO_ISOL_DIS	1b	Allow isolation of port 5 per BMC request. 0b = Enabled 1b = Disabled
12	LAN4_FTCO_ISOL_DIS	1b	Allow isolation of port 4 per BMC request. 0b = Enabled 1b = Disabled
11	Multi-Drop NC-SI	1b	0b = Point-to-point 1b = Multi-drop
10	Reserved	0b	Reserved.
9	EMP_LINK_ON	0b	Copy of the PRTPM_GC.EMP_LINK_ON bit for EMP use. 0b = Disabled 1b = Enabled
8	Redfish Support for PLDM (Type 6)	1b	0b = Disabled 1b = Enabled
7	FW Update over PLDM Support	0b	Support for PLDM base is defined as bit 7 | bit 6 (FW update or monitoring and control support). 0b = Disabled 1b = Enabled
6	PLDM Monitoring and Control	0b	Support for PLDM base is defined as bit 7 | bit 6 (FW update or monitoring and control support). 0b = Disabled 1b = Enabled
5	OEM over MCTP Support	0b	0b = Disabled 1b = Enabled
4	NC-SI over MCTP Support	0b	0b = Disabled 1b = Enabled
3	LAN3_FTCO_ISOL_DIS	1b	Allow isolation of port 3 per BMC request. 0b = Enabled 1b = Disabled
2	LAN2_FTCO_ISOL_DIS	1b	Allow isolation of port 2 per BMC request. 0b = Enabled 1b = Disabled
1	LAN1_FTCO_ISOL_DIS	1b	Allow isolation of port 1 per BMC request. 0b = Enabled 1b = Disabled
0	LAN0_FTCO_ISOL_DIS	1b	Allow isolation of port 0 per BMC request. 0b = Enabled 1b = Disabled

6.3.57.4 PLDM Control Word (0x0003)

Bit(s)	Field Name	Default NVM Value	Description
15:8	Reserved	0x0	
7	Expose Power Sensors	1b	
6	Expose Thermal Trip State Sensors	1b	
5	Expose Configuration and Configuration Change	1b	
4	Expose Presence State	1b	
3	Expose Health State	1b	
2	Expose Thermal Sensors	1b	
1	Expose Link Speed	1b	
0	Expose Link Status	1b	

6.3.57.5 RDE Control Word (0x0004)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Reserved	0x0	Reserved for other schemas including OEM schema extensions.
3	EthernetInterface v1.4.0	1b	Ethernet Interface schema.
2	Reserved	0b	Reserved.
1	VLanNetworkInterface v1.1.1	1b	VLAN Network Interface schema.
0	ACD	1b	ACD collection.

6.3.57.6 OCP NIC Parameters (0x0005)

Bit(s)	Field Name	Default NVM Value	Description
15:7	Reserved	0x0	
6	FAN_ON_AUX Valid	0b	
5	FAN_ON_AUX Value	0b	
4:0	FAN_ON_AUX SDP	0x0	Valid only if FAN_ON_AUX valid = 1.

6.3.57.7 Reserved (0x0006)

6.3.57.8 Sideband Configuration Pointer (0x0007)

This module is l28 bytes and must be mapped in the first valid 4 KB sector of the Flash.

Bit(s)	Field Name	Default NVM Value	Description
15:0	Sideband Configuration Pointer	0xFFFF	Points to Sideband Configuration Structure Section. For Sideband Configuration Structure inner structure, see Section 6.3.58.

6.3.57.9 Reserved (0x0008)

6.3.57.10 Traffic Types Parameters (0x0009)

Bit(s)	Field Name	Default NVM Value	Description
15:14	Port 7 Traffic Types	01b	Defines which type of traffic can flow to and from primary BMC connection on port 3. 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. 00b = Reserved 01b = Network to BMC traffic only 10b = OS2BMC traffic only 11b = Both Network to BMC traffic and OS2BMC traffic
13:12	Port 6 Traffic Types	01b	Defines which type of traffic can flow to and from primary BMC connection on port 3. 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. 00b = Reserved 01b = Network to BMC traffic only 10b = OS2BMC traffic only 11b = Both Network to BMC traffic and OS2BMC traffic
11:10	Port 5 Traffic Types	01b	Defines which type of traffic can flow to and from primary BMC connection on port 2. 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. 00b = Reserved 01b = Network to BMC traffic only 10b = OS2BMC traffic only 11b = Both Network to BMC traffic and OS2BMC traffic
9:8	Port 4 Traffic Types	01b	Defines which type of traffic can flow to and from primary BMC connection on port 2. 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. 00b = Reserved 01b = Network to BMC traffic only 10b = OS2BMC traffic only 11b = Both Network to BMC traffic and OS2BMC traffic
7:6	Port 3 Traffic Types	01b	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. 00b = Reserved 01b = Network to BMC traffic only 10b = OS2BMC traffic only 11b = Both Network to BMC traffic and OS2BMC traffic
5:4	Port 2 Traffic Types	01b	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. 00b = Reserved 01b = Network to BMC traffic only 10b = OS2BMC traffic only 11b = Both Network to BMC traffic and OS2BMC traffic
3:2	Port 1 Traffic Types	01b	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. 00b = Reserved 01b = Network to BMC traffic only 10b = OS2BMC traffic only 11b = Both Network to BMC traffic and OS2BMC traffic

Bit(s)	Field Name	Default NVM Value	Description
1:0	Port 0 Traffic Types	01b	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. 00b = Reserved 01b = Network to BMC traffic only 10b = OS2BMC traffic only 11b = Both Network to BMC traffic and OS2BMC traffic

6.3.57.11 OEM Section Pointer (0x000A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	OEM Section Pointer	0xFFFF	Points to OEM Section Section. For OEM Section inner structure, see Section 6.3.59.

6.3.58 Sideband Configuration Structure Section

This section describes the setting of the different pass-through interfaces (SMBus, NC-SI, and MCTP).

Table 6-64. Sideband Configuration Structure Section Summary Table

Word Offset	Description	Section Reference
0x0000	Section Length	6.3.58.1
0x0001	SMBus Maximum Fragment Size	6.3.58.2
0x0002	SMBus Notification Timeout and Flags	6.3.58.3
0x0003	NC-SI Configuration 1	6.3.58.4
0x0004	NC-SI Configuration 2	6.3.58.5
0x0005	NC-SI Flow Control XOFF	6.3.58.6
0x0006	NC-SI Flow Control XON	6.3.58.7
0x0007	NC-SI HW Arbitration Configuration	6.3.58.8
0x0008 - 0x000C	Reserved	6.3.58.9
0x000D	OEM IANA	6.3.58.10
0x000E	NC-SI over MCTP Message Types	6.3.58.11
0x000F	NC-SI over MCTP Configuration	6.3.58.12
0x0010	MCTP Rate Limiter Config 1	6.3.58.13
0x0011	MCTP Rate Limiter Config 2	6.3.58.14
0x0012	Port to MDEF Mapping 0	6.3.58.15
0x0013	Port to MDEF Mapping 1	6.3.58.16
0x0014	Port to MDEF Mapping 2	6.3.58.17

6.3.58.1 Section Length (0x0000)

The length of the section in words. Note that section length does not include a count for the section length word.

Bit(s)	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 -> Port to MDEF Mapping 2

6.3.58.2 SMBus Maximum Fragment Size (0x0001)

Bit(s)	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).

6.3.58.3 SMBus Notification Timeout and Flags (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:8	SMBus Notification Timeout (ms)	0xFF	
7:6	SMBus Connection Speed	00b	SMBus Connection Speed 00b = Standard SMBus connection 01b = 400 Kb/s fast I^2C. 10b = 1 Mb/s fast+ I^2C. 11b = Reserved.
5	SMBus Block Read Command	0b	SMBus Block Read Command 0b = Block read command is 0xC0. 1b = Block read command is 0xD0.
4	Reserved	0b	Reserved.
3	Enable Fairness Arbitration	1b	MCTP over SMBus feature 0b = Disable fairness arbitration. 1b = Enable fairness arbitration.
2	Disable SMBus ARP Functionality	0b	Disable SMBus ARP Functionality 0b = SMBus ARP Enabled. 1b = SMBus ARP Disabled.
1	SMBus ARP PEC	1b	SMBus ARP PEC 0b = Disable PEC Disable SMBus ARP PEC. 1b = Enable PEC Enable SMBus ARP PEC. Should be set in MCTP modes.
0	SMBus Transaction PEC	0b	SMBus Transactions PEC 0b = 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. 1b = 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. Should be set in MCTP modes.

6.3.58.4 NC-SI Configuration 1 (0x0003)

Bit(s)	Field Name	Default NVM Value	Description
15	NC-SI Channel to Port Mapping Table Valid	0b	0b = Table Invalid Use default algorithm described in Channel ID mapping section. 1b = Table Valid Use the mapping defined in this word.
14	Flow Control	0b	0b = NC-SI flow control disable. 1b = NC-SI flow control enable.
13:10	NC-SI Version	0x1	Supported NC-SI version. Valid values are: 0x0 = NC-SI 1.0.1 0x1 = NC-SI 1.1
9	NC-SI HW Arbitration Enable	0b	0b = Not supported 1b = Supported
8	Reserved		Reserved.
7:5	Package ID	000b	Meaningful only when bit 15 of NC-SI Configuration 2 word (offset 0x07) is cleared.
4:0	Reserved	0x0	Reserved. Must be zero.

6.3.58.5 NC-SI Configuration 2 (0x0004)

Bit(s)	Field Name	Default NVM Value	Description
15:14	Read NC-SI Package ID from SDP	00b	Read NC-SI Package ID. 00b = PackageID from NVM Package ID is read from *NVM Package ID* field in the NC-SI Configuration 01b = PackageID from NVM and SDPs #0 and #1. The package ID is (NVM Package ID[2], SDP#1 value, SDP#0 value). 10b = PackageID from NVM and SDP#1. The package ID is (NVM Package ID[2], SDP#1 value, NVM Package ID[0]). 11b = 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	0x3	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.

6.3.58.6 NC-SI Flow Control XOFF (0x0005)

Bit(s)	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).

6.3.58.7 NC-SI Flow Control XON (0x0006)

Bit(s)	Field Name	Default NVM Value	Description
15:0	XON Threshold	0x1340	Tx buffer watermark 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 + 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).

6.3.58.8 NC-SI HW Arbitration Configuration (0x0007)

Bit(s)	Field Name	Default NVM Value	Description
15:0	TOKEN Timeout	0xA000	NC-SI Hardware Arbitration TOKEN Timeout (in NC-SI REF_CLK cycles - 20 ns). Setting value to 0 disables the timeout mechanism.

6.3.58.9 Reserved (0x0008 - 0x000C)

6.3.58.10 OEM IANA (0x000D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	OEM IANA	0x0	Identifies the OEM targeted by this image. The set of commands accepted depends on the IANA value in this field. The regular Intel OEM commands are accepted only with IANA 0x157.

6.3.58.11 NC-SI over MCTP Message Types (0x000E)

Bit(s)	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.

6.3.58.12 NC-SI over MCTP Configuration (0x000F)

Bit(s)	Field Name	Default NVM Value	Description
15:7	Reserved	0x0	Reserved.
6	Simplified MCTP	0b	If set, only SOM and EOM bits are used for the reassembly process. Relevant only in SMBus mode.
5	Disable ACLs	0b	If set, the ACLs on the PCIe VDMs are disabled.
4:0	Reserved	0x0	Reserved.

6.3.58.13 MCTP Rate Limiter Config 1 (0x0010)

MCTP rate limiter configuration for first channel.

Bit(s)	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 bits. This value provides a bit rate of 1 Mb/s.

6.3.58.14 MCTP Rate Limiter Config 2 (0x0011)

MCTP rate limiter configuration for first channel.

Bit(s)	Field Name	Default NVM Value	Description
15	Decision Point	0b	Defines if, when credits are available, a full MCTP message is sent or a single VDM is sent. 0b = Per VDM 1b = 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).

6.3.58.15 Port to MDEF Mapping 0 (0x0012)

Defines the ports mapping to MDEF sets for sets 0-2.

Bit(s)	Field Name	Default NVM Value	Description
15	Reserved	0b	Reserved.
14:10	MDEF Set 2 Port	0x1F	If 0x1F the MDEF is not assigned to any port.
9:5	MDEF Set 1 Port	0x1F	If 0x1F the MDEF is not assigned to any port.
4:0	MDEF Set 0 Port	0x1F	If 0x1F the MDEF is not assigned to any port.

6.3.58.16 Port to MDEF Mapping 1 (0x0013)

Defines the ports mapping to MDEF sets for sets 3-5.

Bit(s)	Field Name	Default NVM Value	Description
15	Reserved	0b	Reserved.
14:10	MDEF Set 5 Port	0x1F	If 0x1F the MDEF is not assigned to any port.
9:5	MDEF Set 4 Port	0x1F	If 0x1F the MDEF is not assigned to any port.
4:0	MDEF Set 3 Port	0x1F	If 0x1F the MDEF is not assigned to any port.

6.3.58.17 Port to MDEF Mapping 2 (0x0014)

Defines the ports mapping to MDEF sets for sets 6-7.

Bit(s)	Field Name	Default NVM Value	Description
15:10	Reserved	0x0	Reserved.
9:5	MDEF Set 7 Port	0x1F	If 0x1F the MDEF is not assigned to any port.
4:0	MDEF Set 6 Port	0x1F	If 0x1F the MDEF is not assigned to any port.

6.3.59 OEM Section Section

This section is the header of the OEM-specific data identifying the OEM for which this data is defined.

Table 6-65. OEM Section Section Summary Table

Word Offset	Description	Section Reference
0x0000	Section Length	6.3.59.1
0x0001	OEM Header	6.3.59.2
0x0002	Reserved	6.3.59.3

6.3.59.1 Section Length (0x0000)

The length of the section in words. Note that section length does not include a count for the section length word.

Bit(s)	Field Name	Default NVM Value	Description
15:0	Section Length		Length in: 2 Bytes unit - 1 First Section -> Word: OEM Section -> Section Length Last Section -> Word: OEM Section -> Reserved

6.3.59.2 OEM Header (0x0001)

Bit(s)	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.

6.3.59.3 Reserved (0x0002)

6.3.60 Auto-Generated Pointers Module Section

Pointers to Type 1/2 words used by EMP and Software.

Table 6-66. Auto-Generated Pointers Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	Module Length	6.3.60.1
0x0001	Pointer to PFPM_APM Section	6.3.60.2
0x0002	Pointer to PFPM_APM Offset	6.3.60.3
0x0003	Pointer to PRTPM_GC Section	6.3.60.4
0x0004	Pointer to PRTPM_GC Offset	6.3.60.5
0x0005	Pointer to GLPCI_CAPSUP Section	6.3.60.6
0x0006	Pointer to GLPCI_CAPSUP Offset	6.3.60.7
0x0007	Pointer to PRTDCB_FCCFG Section	6.3.60.8
0x0008	Pointer to PRTDCB_FCCFG Offset	6.3.60.9
0x0009	Pointer to PFGEN_PORTNUM Section	6.3.60.10
0x000A	Pointer to PFGEN_PORTNUM Offset	6.3.60.11
0x000B	Pointer to PFPCI_FUNC2 Section	6.3.60.12
0x000C	Pointer to PFPCI_FUNC2 Offset	6.3.60.13
0x000D	Pointer to PF_VT_PFALLOC_PCIE Section	6.3.60.14
0x000E	Pointer to PF_VT_PFALLOC_PCIE Offset	6.3.60.15
0x000F	Pointer to PF_VT_PFALLOC Section	6.3.60.16
0x0010	Pointer to PF_VT_PFALLOC Offset	6.3.60.17
0x0011	Pointer to GLPCI_REVID Section	6.3.60.18
0x0012	Pointer to GLPCI_REVID Offset	6.3.60.19
0x0013	Pointer to PFPCI_DEVID Section	6.3.60.20
0x0014	Pointer to PFPCI_DEVID Offset	6.3.60.21
0x0015	Pointer to GLPCI_SUBVENID Section	6.3.60.22
0x0016	Pointer to GLPCI_SUBVENID Offset	6.3.60.23
0x0017	Pointer to PFPCI_SUBSYSID Section	6.3.60.24
0x0018	Pointer to PFPCI_SUBSYSID Offset	6.3.60.25
0x0019	Pointer to GLPCI_VENDORID Section	6.3.60.26
0x001A	Pointer to GLPCI_VENDORID Offset	6.3.60.27
0x001B	Pointer to PFPCI_FUNC Section	6.3.60.28
0x001C	Pointer to PFPCI_FUNC Offset	6.3.60.29
0x001D	Pointer to PFPCI_CNF Section	6.3.60.30
0x001E	Pointer to PFPCI_CNF Offset	6.3.60.31
0x001F	Pointer to GLPCI_CAPCTRL Section	6.3.60.32
0x0020	Pointer to GLPCI_CAPCTRL Offset	6.3.60.33
0x0021	Pointer to PFGEN_PORTNUM_CAR Section	6.3.60.34

Table 6-66. Auto-Generated Pointers Module Section Summary Table [continued]

Word Offset	Description	Section Reference
0x0022	Pointer to PFGEN_PORTNUM_CAR Offset	6.3.60.35
0x0023	Pointer to GLFOC_CACHE_CTL Section	6.3.60.36
0x0024	Pointer to GLFOC_CACHE_CTL Offset	6.3.60.37
0x0025	Pointer to PRTGEN_CNF Section	6.3.60.38
0x0026	Pointer to PRTGEN_CNF Offset	6.3.60.39
0x0027	Pointer to PF_VT_PFALLOC_HIF Section	6.3.60.40
0x0028	Pointer to PF_VT_PFALLOC_HIF Offset	6.3.60.41
0x0029	Pointer to PRTMAC_HSECTL1 Section	6.3.60.42
0x002A	Pointer to PRTMAC_HSECTL1 Offset	6.3.60.43
0x002B	Pointer to PRT_TDPUL2TAGSEN Section	6.3.60.44
0x002C	Pointer to PRT_TDPUL2TAGSEN Offset	6.3.60.45
0x002D	Pointer to GLGEN_MAC_LINK_TOPO Section	6.3.60.46
0x002E	Pointer to GLGEN_MAC_LINK_TOPO Offset	6.3.60.47

6.3.60.1 Module Length (0x0000)

Bit(s)	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 GLGEN_MAC_LINK_TOPO Offset

6.3.60.2 Pointer to PFPM_APM Section (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to PFPM_APM Section	0x0	

6.3.60.3 Pointer to Pfpm_apm Offset (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to PFPM_APM Offset	0x59	

6.3.60.4 Pointer to PRTPM_GC Section (0x0003)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Cloned Pointer to PRTPM_GC Section		

6.3.60.5 Pointer to PRTPM_GC Offset (0x0004)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to PRTPM_GC Offset	0x7C	

6.3.60.6 Pointer to GLPCI_CAPSUP Section (0x0005)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to GLPCI_CAPSUP Section	0x0	

6.3.60.7 Pointer to GLPCI_CAPSUP Offset (0x0006)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to GLPCI_CAPSUP Offset	0x51	

6.3.60.8 Pointer to PRTDCB_FCCFG Section (0x0007)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to PRTDCB_FCCFG Section	0x0	

6.3.60.9 Pointer to PRTDCB_FCCFG Offset (0x0008)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to PRTDCB_FCCFG Offset	0x35F	

6.3.60.10 Pointer to PFGEN_PORTNUM Section (0x0009)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to PFGEN_PORTNUM Section	0x0	

6.3.60.11 Pointer to PFGEN_PORTNUM Offset (0x000A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to PFGEN_PORTNUM Offset	0x40B	

6.3.60.12 Pointer to PFPCI_FUNC2 Section (0x000B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Cloned Pointer to PFINT_ALLOC_PCI Section		

6.3.60.13 Pointer to PFPCI_FUNC2 Offset (0x000C)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to PFINT_ALLOC_PCI Offset	0x3	

6.3.60.14 Pointer to PF_VT_PFALLOC_PCIE Section (0x000D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Cloned Pointer to PF_VT_PFALLOC_PCIE Section		

6.3.60.15 Pointer to PF_VT_PFALLOC_PCIE Offset (0x000E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to PF_VT_PFALLOC_PCIE Offset	0x77	

6.3.60.16 Pointer to PF_VT_PFALLOC Section (0x000F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Cloned Pointer to PF_VT_PFALLOC Section		

6.3.60.17 Pointer to PF_VT_PFALLOC Offset (0x0010)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to PF_VT_PFALLOC Offset	0x41E	

6.3.60.18 Pointer to GLPCI_REVID Section (0x0011)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to GLPCI_REVID Section	0x0	Points to PCIR Type 1/2 Section. For PCIR Type 1/2 inner structure, see Section 6.3.47.

6.3.60.19 Pointer to GLPCI_REVID Offset (0x0012)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to GLPCI_REVID Offset	0x9	

6.3.60.20 Pointer to PFPCI_DEVID Section (0x0013)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Cloned Pointer to PFPCI_DEVID Section		

6.3.60.21 Pointer to PFPCI_DEVID Offset (0x0014)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to PFPCI_DEVID Offset	0x3C	

6.3.60.22 Pointer to GLPCI_SUBVENID Section (0x0015)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Cloned Pointer to GLPCI_SUBVENID Section		

6.3.60.23 Pointer to GLPCI_SUBVENID Offset (0x0016)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to GLPCI_SUBVENID Offset	0x5B	

6.3.60.24 Pointer to PFPCI_SUBSYSID Section (0x017)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Cloned Pointer to PFPCI_SUBSYSID Section		

6.3.60.25 Pointer to PFPCI_SUBSYSID Offset (0x0018)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to PFPCI_SUBSYSID Offset	0x16	

6.3.60.26 Pointer to GLPCI_VENDORID Section (0x0019)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Cloned Pointer to GLPCI_VENDORID Section		

6.3.60.27 Pointer to GLPCI_VENDORID Offset (0x001A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to GLPCI_VENDORID Offset	0x57	

6.3.60.28 Pointer to PFPCI_FUNC Section (0x001B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Cloned Pointer to PFPCI_FUNC Section		

6.3.60.29 Pointer to PFPCI_FUNC Offset (0x000C)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to PFPCI_FUNC Offset	0x14	

6.3.60.30 Pointer to PFPCI_CNF Section (0x000D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Cloned Pointer to PFPCI_CNF Section		

6.3.60.31 Pointer to PFPCI_CNF Offset (0x000E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to PFPCI_CNF Offset	0x60	

6.3.60.32 Pointer to GLPCI_CAPCTRL Section (0x000F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Cloned Pointer to GLPCI_CAPCTRL Section		

6.3.60.33 Pointer to GLPCI_CAPCTRL Offset (0x0020)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to GLPCI_CAPCTRL Offset	0x4F	

6.3.60.34 Pointer to PFGEN_PORTNUM_CAR Section (0x0021)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Cloned Pointer to PFGEN_PORTNUM_CAR Section		

6.3.60.35 Pointer to PFGEN_PORTNUM_CAR Offset (0x0022)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to PFGEN_PORTNUM_CAR Offset	0x46	

6.3.60.36 Pointer to GLFOC_CACHE_CTL Section (0x0023)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Cloned Pointer to GLFOC_CACHE_CTL Section		

6.3.60.37 Pointer to GLFOC_CACHE_CTL Offset (0x0024)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to GLFOC_CACHE_CTL Offset	0x1A5	

6.3.60.38 Pointer to PRTGEN_CNF Section (0x0025)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Cloned Pointer to PRTGEN_CNF Section		

6.3.60.39 Pointer to PRTGEN_CNF Offset (0x0026)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to PRTGEN_CNF Offset	0x6C	

6.3.60.40 Pointer to PF_VT_PFALLOC_HIF Section (0x0027)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Cloned Pointer to PF_VT_PFALLOC_HIF Section		

6.3.60.41 Pointer to PF_VT_PFALLOC_HIF Offset (0x0028)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to PF_VT_PFALLOC_HIF Offset	0x29	

6.3.60.42 Pointer to PRTMAC_HSECTL1 Section (0x0029)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Cloned Pointer to PRTMAC_HSECTL1 Section		

6.3.60.43 Pointer to PRTMAC_HSECTL1 Offset (0x002A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to PRTMAC_HSECTL1 Offset	0x2B2	

6.3.60.44 Pointer to PRT_TDPUL2TAGSEN Section (0x002B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Cloned Pointer to PRT_TDPUL2TAGSEN Section		

6.3.60.45 Pointer to PRT_TDPUL2TAGSEN Offset (0x002C)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to PRT_TDPUL2TAGSEN Offset	0x3	

6.3.60.46 Pointer to GLGEN_MAC_LINK_TOPO Section (0x002D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Cloned Pointer to GLGEN_MAC_LINK_TOPO Section		

6.3.60.47 Pointer to GLGEN_MAC_LINK_TOPO Offset (0x002E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Pointer to GLGEN_MAC_LINK_TOPO Offset	0x2	

6.3.61 NVM Image CSS Header Section

Table 6-67. NVM Image CSS Header Section Summary Table

Word Offset	Description	Section Reference
0x0000	moduleTypeL	6.3.61.1
0x0001	moduleTypeH	6.3.61.2
0x0002	headerLenL	6.3.61.3
0x0003	headerLenH	6.3.61.4
0x0004	headerVersionL	6.3.61.5
0x0005	headerVersionH	6.3.61.6
0x0006	moduleIDL	6.3.61.7
0x0007	moduleIDH	6.3.61.8
0x0008	moduleVendorL	6.3.61.9
0x0009	moduleVendorH	6.3.61.10
0x000A	dateL	6.3.61.11
0x000B	dateH	6.3.61.12
0x000C	sizeL	6.3.61.13
0x000D	sizeH	6.3.61.14
0x000E	keySizeL	6.3.61.15
0x000F	keySizeH	6.3.61.16
0x0010	modulusSizeL	6.3.61.17
0x0011	modulusSizeH	6.3.61.18
0x0012	exponentSizeL	6.3.61.19
0x0013	exponentSizeH	6.3.61.20
0x0014	lad_srevL	6.3.61.21
0x0015	lad_srevH	6.3.61.22
0x0016 - 0x0017	Reserved	6.3.61.23
0x0018	lad_fw_entry_offsetL	6.3.61.24
0x0019	lad_fw_entry_offsetH	6.3.61.25
0x001A - 0x001B	Reserved	6.3.61.26
0x001C	lad_image_unique_idL	6.3.61.27
0x001D	lad_image_unique_idH	6.3.61.28
0x001E	lad_module_idL	6.3.61.29
0x001F	lad_module_idH	6.3.61.30
0x0020	Reserved	6.3.61.31

6.3.61.1 moduleTypeL (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	moduleTypeL	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	0xA1	

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	1b	
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)

Bits	Field Name	Default NVM Value	Description
15:0	DateL	0x20130530	0xMMDD

6.3.61.12 dateH (0x000B)

Bits	Field Name	Default NVM Value	Description
15:0	DateH		0xYYYY

6.3.61.13 sizeL (0x000C)

Bits	Field Name	Default NVM Value	Description
15:0	sizeL	0x000A0000	

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	0x40	

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	0x40	

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	0x0	

6.3.61.22 lad_srevH (0x0015)

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevH		

6.3.61.23 Reserved (0x0016 - 0x0017)

6.3.61.24 lad_fw_entry_offsetL (0x0018)

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetL	0x102C0	

6.3.61.25 lad_fw_entry_offsetL (0x0019)

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetH		

6.3.61.26 Reserved (0x001A - 0x001B)

6.3.61.27 lad_image_unique_idL (0x001C)

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idL		

6.3.61.28 lad_image_unique_idH (0x001D)

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idH		

6.3.61.29 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 Mini-loader

6.3.61.30 lad_module_idH (0x001F)

Bits	Field Name	Default NVM Value	Description
15	Reserved	0b	Reserved.
14:1	lad_module_idH		

6.3.61.31 Reserved (0x0020)

6.3.62 NVM Key and Signature Section

Table 6-68. NVM Key and Signature Section Summary Table

Word Offset	Description	Section Reference
0x0000 + 1*n, n=0...127	RSA Public Key	6.3.62.1
0x0080	RSA ExponentL	6.3.62.2
0x0081	RSA ExponentH	6.3.62.3
0x0082 + 1*n, n=0...127	Encrypted SHA256 Hash	6.3.62.4

6.3.62.1 RSA Public Key[n] (0x0000 + 1*n, n=0...127)

Bit(s)	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.3.62.2 RSA ExponentL (0x0080)

Bit(s)	Field Name	Default NVM Value	Description
15:0	RSA ExponentL	0x0	

6.3.62.3 RSA ExponentH (0x0081)

Bit(s)	Field Name	Default NVM Value	Description
15:0	RSA ExponentH	0x0	

6.3.62.4 Encrypted SHA256 Hash[n] (0x0082 + 1*n, n=0...127)

Bit(s)	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.3.63 NVM Image Auth Header Section

Table 6-69. NVM Image Auth Header Section Summary Table

Word Offset	Description	Section Reference
0x0000	Device Blank NVM Device ID	6.3.63.1
0x0001	Max Module AreaL	6.3.63.2
0x0002	Max Module AreaH	6.3.63.3
0x0003	Current Module AreaL	6.3.63.4
0x0004	Current Module AreaH	6.3.63.5
0x0005	Reserved	6.3.63.6
0x0006	Code Revision	6.3.63.7
0x0007	Reserved Spare Word	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	0x374C	

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	Reserved.

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 SR1 - Should Be Copy of Shadow RAM: Section Clone

Table 6-70. SR1 - Should Be Copy of Shadow RAM: Section Clone Summary Table

Word Offset	Description	Section Reference
0x10160	SR1	6.3.64.1

6.3.64.1 SR1 (0x10160)

Raw data module length: variable

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.

6.3.65 ML CSS Header Section

Table 6-71. ML CSS Header Section Summary Table

Word Offset	Description	Section Reference
0x0000	moduleTypeL	6.3.65.1
0x0001	moduleTypeH	6.3.65.2
0x0002	headerLenL	6.3.65.3
0x0003	headerLenH	6.3.65.4
0x0004	headerVersionL	6.3.65.5
0x0005	headerVersionH	6.3.65.6
0x0006	moduleIDL	6.3.65.7
0x0007	moduleIDH	6.3.65.8
0x0008	moduleVendorL	6.3.65.9
0x0009	moduleVendorH	6.3.65.10
0x000A	dateL	6.3.65.11
0x000B	dateH	6.3.65.12
0x000C	sizeL	6.3.65.13
0x000D	sizeH	6.3.65.14
0x000E	keySizeL	6.3.65.15
0x000F	keySizeH	6.3.65.16
0x0010	modulusSizeL	6.3.65.17
0x0011	modulusSizeH	6.3.65.18
0x0012	exponentSizeL	6.3.65.19
0x0013	exponentSizeH	6.3.65.20
0x0014	lad_srevL	6.3.65.21
0x0015	lad_srevH	6.3.65.22
0x0016 - 0x0017	Reserved	6.3.65.23
0x0018	lad_fw_entry_offsetL	6.3.65.24
0x0019	lad_fw_entry_offsetH	6.3.65.25
0x001A - 0x001B	Reserved	6.3.65.26
0x001C	lad_image_unique_idL	6.3.65.27
0x001D	lad_image_unique_idH	6.3.65.28
0x001E	lad_module_idL	6.3.65.29
0x001F	lad_module_idH	6.3.65.30
0x0020 + 1*n, n=0...31	Reserved	6.3.65.31

6.3.65.1 moduleTypeL (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	moduleTypeL	0x6	

6.3.65.2 moduleTypeH (0x0001)

Bits	Field Name	Default NVM Value	Description
15:0	moduleTypeH		

6.3.65.3 headerLenL (0x0002)

Bits	Field Name	Default NVM Value	Description
15:0	headerLenL	0xA1	

6.3.65.4 headerLenH (0x0003)

Bits	Field Name	Default NVM Value	Description
15:0	headerLenH		

6.3.65.5 headerVersionL (0x0004)

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionL	0x00010000	

6.3.65.6 headerVersionH (0x0005)

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionH		

6.3.65.7 moduleIDL (0x0006)

Bits	Field Name	Default NVM Value	Description
15:0	moduleIDL	0x0	

6.3.65.8 moduleIDH (0x0007)

Bits	Field Name	Default NVM Value	Description
15	signMode	1b	
14:0	moduleIDH		

6.3.65.9 moduleVendorL (0x0008)

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorL	0x00008086	

6.3.65.10 moduleVendorH (0x0009)

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorH		

6.3.65.11 dateL (0x000A)

Bits	Field Name	Default NVM Value	Description
15:0	DateL	0x20130530	0xMMDD

6.3.65.12 dateH (0x000B)

Bits	Field Name	Default NVM Value	Description
15:0	DateH		0xYYYY

6.3.65.13 sizeL (0x000C)

Bits	Field Name	Default NVM Value	Description
15:0	sizeL	0x0	

6.3.65.14 sizeH (0x000D)

Bits	Field Name	Default NVM Value	Description
15:0	sizeH		

6.3.65.15 keySizeL (0x000E)

Bits	Field Name	Default NVM Value	Description
15:0	keySizeL	0x40	

6.3.65.16 keySizeH (0x000F)

Bits	Field Name	Default NVM Value	Description
15:0	keySizeH		

6.3.65.17 modulusSizeL (0x0010)

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeL	0x40	

6.3.65.18 modulusSizeH (0x0011)

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeH		

6.3.65.19 exponentSizeL (0x0012)

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeL	0x1	

6.3.65.20 exponentSizeH (0x0013)

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeH		

6.3.65.21 lad_srevL (0x0014)

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevL	0x0	

6.3.65.22 lad_srevH (0x0015)

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevH		

6.3.65.23 Reserved (0x0016 - 0x0017)

6.3.65.24 lad_fw_entry_offsetL (0x0018)

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetL	0x2C0	

6.3.65.25 lad_fw_entry_offsetL (0x0019)

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetH		

6.3.65.26 Reserved (0x001A - 0x001B)

6.3.65.27 lad_image_unique_idL (0x001C)

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idL	0x0	

6.3.65.28 lad_image_unique_idH (0x001D)

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idH	0x0	

6.3.65.29 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 Mini-loader

6.3.65.30 lad_module_idH (0x001F)

Bits	Field Name	Default NVM Value	Description
15	Reserved	0b	Reserved.
14:1	lad_module_idH		

6.3.65.31 Reserved[n] (0x0020 + 1*n, n=0...31)

6.3.66 ML Key and Signature Section

Table 6-72. ML Key and Signature Section Summary Table

Word Offset	Description	Section Reference
0x0000 + 1*n, n=0...127	RSA Public Key	6.3.66.1
0x0080	RSA ExponentL	6.3.66.2
0x0081	RSA ExponentH	6.3.66.3
0x0082 + 1*n, n=0...127	Encrypted SHA256 Hash	6.3.66.4

6.3.66.1 RSA Public Key[n] (0x0000 + 1*n, n=0...127)

Bit(s)	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.3.66.2 RSA ExponentL (0x0080)

Bit(s)	Field Name	Default NVM Value	Description
15:0	RSA ExponentL	0x0	

6.3.66.3 RSA ExponentH (0x0081)

Bit(s)	Field Name	Default NVM Value	Description
15:0	RSA ExponentH	0x0	

6.3.66.4 Encrypted SHA256 Hash[n] (0x0082 + 1*n, n=0...127)

Bit(s)	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.3.67 ML Key and Signature - Signed Section

6.3.68 ML Auth Header Section

Table 6-73. ML Auth Header Section Summary Table

Word Offset	Description	Section Reference
0x0000	Device Blank NVM Device ID	6.3.68.1
0x0001	Max Module AreaL	6.3.68.2
0x0002	Max Module AreaH	6.3.68.3
0x0003	Current Module AreaL	6.3.68.4
0x0004	Current Module AreaH	6.3.68.5
0x0005	Reserved	6.3.68.6
0x0006	Code Revision	6.3.68.7
0x0007	Reserved Spare Word	6.3.68.8

6.3.68.1 Device Blank NVM Device ID (0x0000)

Bits	Field Name	Default NVM Value	Description
15:0	Device Blank NVM Device ID	0x1888	

6.3.68.2 Max Module AreaL (0x0001)

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaL	0x1000	

6.3.68.3 Max Module AreaH (0x0002)

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaH	0x0009	

6.3.68.4 Current Module AreaL (0x0003)

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaL	0xE000	

6.3.68.5 Current Module AreaH (0x0004)

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaH	0x0006	

6.3.68.6 Reserved (0x0005)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.68.7 Code Revision (0x0006)

Bits	Field Name	Default NVM Value	Description
15:8	Major Revision	0x0	
7:0	Minor Revision	0x0	

6.3.68.8 Reserved Spare Word (0x0007)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved Spare Word	0x0	

6.3.69 Extended ML Header Section

Table 6-74. Extended ML Header Section Summary Table

Word Offset	Description	Section Reference
0x0000	Reserved	6.3.69.1
0x0001	Analog PHY pre PLL Configuration Pointer	6.3.69.2
0x0002	CSR Protected List Pointer	6.3.69.3
0x0003	PCIe Analog Pointer	6.3.69.4
0x0004	PCIR Fixed Auto-Load Pointer	6.3.69.5
0x0005	POR Fixed Auto-Load Pointer	6.3.69.6
0x0006	PCIR PFA Auto-Load Whitelist Pointer	6.3.69.7
0x0007	POR PFA Auto-Load Whitelist Pointer	6.3.69.8
0x0008	Reserved	6.3.69.9
0x0009	1st NVM Bank Pointer	6.3.69.10
0x000A	NVM Bank Area Size	6.3.69.11
0x000B	1st OROM Bank Pointer	6.3.69.12
0x000C	OROM Bank Area Size	6.3.69.13
0x000D	1st TLV Extension Bank Pointer	6.3.69.14
0x000E	TLV Extension Bank Area Size	6.3.69.15
0x000F	Preserved Field Area Pointer	6.3.69.16
0x0010	Recovery Firmware Pointer	6.3.69.17
0x0011	Reserved	6.3.69.18
0x0012	Factory Settings Size	6.3.69.19
0x0013	Last PFA Word Pointer	6.3.69.20
0x0014	LVK Hashes Pointer	6.3.69.21
0x0015	Reserved	6.3.69.22

6.3.69.1 Reserved (0x0000)

6.3.69.2 Analog PHY pre PLL Configuration Pointer (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Analog PLL Configuration Pointer	0x0	Points to Analog PHY pre PLL Configuration Section. For Analog PHY pre PLL Configuration inner structure, see Section 6.3.71.

6.3.69.3 CSR Protected List Pointer (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:0	CSR Protected List Pointer	0x0	Points to CSR Protected List Section. For CSR Protected List inner structure, see Section 6.3.72.

6.3.69.4 PCIe Analog Pointer (0x0003)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PCIe Analog Pointer	0xFFF	Points to PCIe Analog Module Section. For PCIe Analog Module inner structure, see Section 6.3.73.

6.3.69.5 PCIR Fixed Auto-Load Pointer (0x0004)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PCIR Fixed Autoload Pointer	0x0	Points to PCIR Registers Auto-Load Module Section. For PCIR Registers Auto-Load Module inner structure, see Section 6.3.74.

6.3.69.6 POR Fixed Auto-Load Pointer (0x0005)

Bit(s)	Field Name	Default NVM Value	Description
15:0	POR Fixed Autoload Pointer	0x0	Points to POR Registers Auto-Load Module Section. For POR Registers Auto-Load Module inner structure, see Section 6.3.75.

6.3.69.7 PCIR PFA Auto-Load Whitelist Pointer (0x0006)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PCIR PFA Autoload Whitelist Pointer	0x0	Points to POR PCIR_PFA Auto-Load Whitelist Module Section. For PCIR_PFA Auto-Load Whitelist Module inner structure, see Section 6.3.76.

6.3.69.8 POR PFA Auto-Load Whitelist Pointer (0x0007)

Bit(s)	Field Name	Default NVM Value	Description
15:0	POR PFA Autoload Whitelist Pointer	0x0	Points to POR POR_PFA Auto-Load Whitelist Module Section. For POR_PFA Auto-Load Whitelist Module inner structure, see Section 6.3.77.

6.3.69.9 Reserved (0x0008)

6.3.69.10 1st NVM Bank Pointer (0x0009)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type		Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	1st NVM Bank Pointer		

6.3.69.11 NVM Bank Area Size (0x000A)

Bit(s)	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	Reserved.
11:0	NVM Bank Area Size		Size expressed in 4 KB sector units.

6.3.69.12 1st OROM Bank Pointer (0x000B)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type		Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	1st OROM Bank Pointer		Pointer to first OROM section (not swapped when section changes).

6.3.69.13 OROM Bank Area Size (0x000C)

Bit(s)	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	Reserved.
11:0	OROM Bank Area Size		Size expressed in 4 KB sector units.

6.3.69.14 1st TLV Extension Bank Pointer (0x000D)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type		Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	1st TLV Extension Bank Pointer		Pointer to first OROM section (not swapped when section changes).

6.3.69.15 TLV Extension Bank Area Size (0x000E)

Bit(s)	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	Reserved.
11:0	TLV Extension Bank Area Size		Size expressed in 4 KB sector units.

6.3.69.16 Preserved Field Area Pointer (0x000F)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type		Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	PFA Pointer		

6.3.69.17 Recovery Firmware Pointer (0x0010)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type		Pointer Type: 0b = Word units. 1b = 4 KB sector units.
14:0	Recovery Firmware Pointer		Pointer to Recovery Firmware section.

6.3.69.18 Reserved (0x0011)

6.3.69.19 Factory Settings Size (0x0012)

Bit(s)	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	Reserved.
11:0	Factory Settings Size		Size expressed in 4 KB sector units.

6.3.69.20 Last PFA Word Pointer (0x0013)

Bit(s)	Field Name	Default NVM Value	Description
15	Pointer Type		Pointer Type: 0b = Word units. 1b = 4 KB sector units. Only the 4 KB sector unit is supported for this pointer.
14:0	Last PFA Word Pointer		Pointer to the end of the PFA excluding the PCI ALT module.

6.3.69.21 LVK Hashes Pointer (0x0014)

Bit(s)	Field Name	Default NVM Value	Description
15:0	LVK Hashes Pointer	0x0	Relative pointer to Low Value Keys Hashes. Points to LVK Hashes Section. For LVK Hashes inner structure See Section 6.3.78.

6.3.69.22 Reserved (0x0015)

6.3.70 ML Image Section

This module is the whole module that has been signed by CSS.

It is in the NVM map for EM tool needs only.

6.3.71 Analog PHY pre PLL Configuration Section

Includes static configurations of PHY Core registers for PLL lock.

Table 6-75. Analog PHY pre PLL Configuration Section Summary Table

Word Offset	Description	Section Reference
0x0000	Section Length	6.3.71.1
0x0001	Reg Write Indirect List	6.3.71.2

6.3.71.1 Section Length (0x0000)

Bit(s)	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

6.3.71.2 Reg Write Indirect List (0x0001)

Raw data module length: variable

6.3.72 CSR Protected List Section

Defines the list of the protected CSRs and their default settings. These registers are made RO to the host as they contain settings that are critical for the host to Flash access when in blank Flash programming mode.

Table 6-76. CSR Protected List Section Summary Table

Word Offset	Description	Section Reference
0x0000	Module Length	6.3.72.1
0x0001 - 0x000C	Reserved	6.3.72.2
0x000D - 0x0010	NVM contents for GLGEN_STAT	6.3.72.3
0x0011 - 0x0014	NVM contents for GLNVM_ALTIMERS	6.3.72.4
0x0015 - 0x0018	Reserved	6.3.72.5
0x0019 - 0x0093	NVM contents for GLNVM_PROTCSR	6.3.72.6

6.3.72.1 Module Length (0x0000)

Bit(s)	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 -> Starting Address Low at GLNVM_PROTCSR[0]

6.3.72.2 Reserved (0x0001 - 0x000C)

6.3.72.3 GLGEN_STAT (0x000D - 0x0010)

6.3.72.3.1 Address Low at GLGEN_STAT (0x000D)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLGEN_STAT	0xB612C	
3:0	Type	0x1	

6.3.72.3.2 Address High at GLGEN_STAT (0x000E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLGEN_STAT		

6.3.72.3.3 Data Low of GLGEN_STAT (0x000F)

6.3.72.3.4 Data High of GLGEN_STAT (0x0010)

6.3.72.4 GLNVM_ALTIMERS (0x0011 - 0x0014)

6.3.72.4.1 Address Low at GLNVM_ALTIMERS (0x0011)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLNVM_ALTIMERS	0xB6140	
3:0	Type	0x1	

6.3.72.4.2 Address High at GLNVM_ALTIMERS (0x0012)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLNVM_ALTIMERS		

6.3.72.4.3 Data Low of GLNVM_ALTIMERS (0x0013)

6.3.72.4.4 Data High of GLNVM_ALTIMERS (0x0014)

6.3.72.5 Reserved (0x0015 - 0x0018)

6.3.72.6 GLNVM_PROTCSR (0x0019 - 0x0093)

6.3.72.6.1 Starting Address Low at GLNVM_PROTCSR (0x0019)

Bit(s)	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLNVM_PROTCSR	0xB6010	
3:0	Type	0x2	

6.3.72.6.2 Starting Address High at GLNVM_PROTCSR (0x001A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLNVM_PROTCSR		

6.3.72.6.3 Attributes at GLNVM_PROTCSR (0x001B)

Bit(s)	Field Name	Default NVM Value	Description
15:5	Length	0x3C	
4:3	Skip	00b	
2:0	Width	000b	

6.3.72.6.4 Data Low of GLNVM_PROTCSR[n] (0x001C + 2*n, n=0...59)

6.3.72.6.5 Data High of GLNVM_PROTCSR[n] (0x001D + 2*n, n=0...59)

6.3.73 PCIe Analog Module Section

Contains read-only parameters that configure the PCIe Transaction layer.

Table 6-77. PCIe Analog Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	Module Length	6.3.73.1
0x0001	PCIe Analog Data	6.3.73.2

6.3.73.1 Module Length (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Module Length		

6.3.73.2 PCIe Analog Data (0x0001)

Raw data module length: variable

This word must be disabled at the image level.

6.3.74 PCIR Registers Auto-Load Module Section

Default setup to registers and internal memories that load on PCIR events.

Table 6-78. PCIR Registers Auto-Load Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	Module Length	6.3.74.1
0x0001	PCIR Auto-Load Duplicate	6.3.74.2
0x0002 - 0x0005	Reserved	6.3.74.3

6.3.74.1 Module Length (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 1 First Section -> Word: PCIR Registers Auto-load Module -> Module Length Last Section -> Word: POR Registers Auto-load Module -> Reserved

6.3.74.2 PCIR Auto-Load Duplicate (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PCIR Autoload Duplicate		

6.3.74.3 Reserved (0x0002 - 0x0005)

6.3.75 POR Registers Auto-Load Module Section

Default setup to registers that load on POR events.

Table 6-79. POR Registers Auto-Load Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	Module Length	6.3.75.1
0x0001	POR Auto-Load Duplicate	6.3.75.2

6.3.75.1 Module Length (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 1 First Section -> Word: POR Registers Auto-load Module -> Module Length Last Section -> Word: POR Registers Auto-load Module -> POR Auto-Load Duplicate

6.3.75.2 POR Auto-Load Duplicate (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:0	POR Autoload Duplicate		

6.3.76 PCIR_PFA Auto-Load Whitelist Module Section

Default setup to registers and internal memories that load on PCIR events.

Table 6-80. PCIR_PFA Auto-Load Whitelist Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	Module Length	6.3.76.1
0x0001	PCIR_PFA Auto-Load Duplicate	6.3.76.2

6.3.76.1 Module Length (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 1 First Section -> Word: PCIR_PFA Auto-load Whitelist Module -> Module Length Last Section -> Word: PCIR_PFA Auto-load Whitelist Module -> PCIR_PFA Auto-Load Duplicate

6.3.76.2 PCIR_PFA Auto-Load Duplicate (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PCIR_PFA Autoload Duplicate		

6.3.77 POR_PFA Auto-Load Whitelist Module Section

Default setup to registers and internal memories that load on POR events.

Table 6-81. POR_PFA Auto-Load Whitelist Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	Module Length	6.3.77.1
0x0001	POR_PFA Auto-Load Duplicate	6.3.77.2

6.3.77.1 Module Length (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 1 First Section -> Word: POR_PFA Auto-load Whitelist Module -> Module Length Last Section -> Word: POR_PFA Auto-load Whitelist Module -> POR_PFA Auto-Load Duplicate

6.3.77.2 POR_PFA Auto-Load Duplicate (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:0	POR_PFA Autoload Duplicate		

6.3.78 LVK Hashes Section

Hashes for Low Value Keys.

Table 6-82. LVK Hashes Section Summary Table

Word Offset	Description	Section Reference
0x0000	Length	6.3.78.1
0x0001 + 1*n, n=0...15	NVM Bank Key Hash	6.3.78.2
0x0011 + 1*n, n=0...15	OS Package Key Hash	6.3.78.3
0x0021 + 1*n, n=0...15	OROM Key Hash	6.3.78.4

6.3.78.1 Length (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 1 First Section -> Word: LVK Hashes -> Length Last Section -> Word: LVK Hashes -> OROM Key Hash

6.3.78.2 NVM Bank Key Hash[n] (0x0001 + 1*n, n=0...15)

Bit(s)	Field Name	Default NVM Value	Description
15:0	NVM Bank Key Hash	0x0	Hash for NVM Bank and Recovery section key.

6.3.78.3 OS Package Key Hash[n] (0x0011 + 1*n, n=0...15)

Bit(s)	Field Name	Default NVM Value	Description
15:0	OS Package Key Hash	0x0	Hash for OS Package keys.

6.3.78.4 OROM Key Hash[n] (0x0021 + 1*n, n=0...15)

Bit(s)	Field Name	Default NVM Value	Description
15:0	OROM Key Hash	0x0	Hash for OROM key.

6.3.79 Recovery FW CSS Header Section

Table 6-83. Recovery FW CSS Header Section Summary Table

Word Offset	Description	Section Reference
0x21800	moduleTypeL	6.3.79.1
0x21801	moduleTypeH	6.3.79.2
0x21802	headerLenL	6.3.79.3
0x21803	headerLenH	6.3.79.4
0x21804	headerVersionL	6.3.79.5
0x21805	headerVersionH	6.3.79.6
0x21806	moduleIDL	6.3.79.7
0x21807	moduleIDH	6.3.79.8
0x21808	moduleVendorL	6.3.79.9
0x21809	moduleVendorH	6.3.79.10
0x2180A	dateL	6.3.79.11
0x2180B	dateH	6.3.79.12
0x2180C	sizeL	6.3.79.13
0x2180D	sizeH	6.3.79.14
0x2180E	keySizeL	6.3.79.15
0x2180F	keySizeH	6.3.79.16
0x21810	modulusSizeL	6.3.79.17
0x21811	modulusSizeH	6.3.79.18
0x21812	exponentSizeL	6.3.79.19
0x21813	exponentSizeH	6.3.79.20
0x21814	lad_srevL	6.3.79.21
0x21815	lad_srevH	6.3.79.22
0x21816 - 0x21817	Reserved	6.3.79.23
0x21818	lad_fw_entry_offsetL	6.3.79.24
0x21819	lad_fw_entry_offsetH	6.3.79.25
0x2181A - 0x2181B	Reserved	6.3.79.26
0x2181C	lad_image_unique_idL	6.3.79.27
0x2181D	lad_image_unique_idH	6.3.79.28
0x2181E	lad_module_idL	6.3.79.29
0x2181F	lad_module_idH	6.3.79.30
0x21820 + 1*n, n=0...31	Reserved	6.3.79.31

6.3.79.1 moduleTypeL (0x21800)

Bits	Field Name	Default NVM Value	Description
15:0	moduleTypeL	0x6	

6.3.79.2 moduleTypeH (0x21801)

Bits	Field Name	Default NVM Value	Description
15:0	moduleTypeH		

6.3.79.3 headerLenL (0x21802)

Bits	Field Name	Default NVM Value	Description
15:0	headerLenL	0xA1	

6.3.79.4 headerLenH (0x21803)

Bits	Field Name	Default NVM Value	Description
15:0	headerLenH		

6.3.79.5 headerVersionL (0x21804)

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionL	0x00010000	

6.3.79.6 headerVersionH (0x21805)

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionH		

6.3.79.7 moduleIDL (0x21806)

Bits	Field Name	Default NVM Value	Description
15:0	moduleIDL	0x0	

6.3.79.8 moduleIDH (0x21807)

Bits	Field Name	Default NVM Value	Description
15	signMode	1b	
14:0	moduleIDH		

6.3.79.9 moduleVendorL (0x21808)

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorL	0x00008086	

6.3.79.10 moduleVendorH (0x21809)

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorH		

6.3.79.11 dateL (0x2180A)

Bits	Field Name	Default NVM Value	Description
15:0	DateL	0x20130530	0xMMDD

6.3.79.12 dateH (0x2180B)

Bits	Field Name	Default NVM Value	Description
15:0	DateH		0xYYYY

6.3.79.13 sizeL (0x2180C)

Bits	Field Name	Default NVM Value	Description
15:0	sizeL	0x0000FA00	

6.3.79.14 sizeH (0x2180D)

Bits	Field Name	Default NVM Value	Description
15:0	sizeH		

6.3.79.15 keySizeL (0x2180E)

Bits	Field Name	Default NVM Value	Description
15:0	keySizeL	0x40	

6.3.79.16 keySizeH (0x2180F)

Bits	Field Name	Default NVM Value	Description
15:0	keySizeH		

6.3.79.17 modulusSizeL (0x21810)

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeL	0x40	

6.3.79.18 modulusSizeH (0x21811)

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeH		

6.3.79.19 exponentSizeL (0x21812)

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeL	0x1	

6.3.79.20 exponentSizeH (0x21813)

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeH		

6.3.79.21 lad_srevL (0x21814)

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevL	0x0	

6.3.79.22 lad_srevH (0x21815)

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevH		

6.3.79.23 Reserved (0x21816 - 0x21817)

6.3.79.24 lad_fw_entry_offsetL (0x21818)

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetL	0x2C0	

6.3.79.25 lad_fw_entry_offsetL (0x21819)

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetH		

6.3.79.26 Reserved (0x2181A - 0x2181B)

6.3.79.27 lad_image_unique_idL (0x2181C)

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idL		

6.3.79.28 lad_image_unique_idH (0x2181D)

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idH		

6.3.79.29 lad_module_idL (0x2181E)

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 0x6= NVM Bank 0x7= Extended Mini-loader 0xB = Recovery Firmware

6.3.79.30 lad_module_idH (0x2181F)

Bits	Field Name	Default NVM Value	Description
15	Reserved	0b	Reserved.
14:1	lad_module_idH		

6.3.79.31 Reserved[n] (0x21820 + 1*n, n=0...31)

6.3.80 Recovery FW Key and Signature Section

Table 6-84. Recovery FW Key and Signature Section Summary Table

Word Offset	Description	Section Reference
0x0000 + 1*n, n=0...127	RSA Public Key	6.3.80.1
0x0080	RSA ExponentL	6.3.80.2
0x0081	RSA ExponentH	6.3.80.3
0x0082 + 1*n, n=0...127	Encrypted SHA256 Hash	6.3.80.4

6.3.80.1 RSA Public Key [n] (0x0000 + 1*n, n=0...127)

Bit(s)	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.3.80.2 RSA ExponentL (0x0080)

Bit(s)	Field Name	Default NVM Value	Description
15:0	RSA ExponentL	0x0	

6.3.80.3 RSA ExponentH (0x0081)

Bit(s)	Field Name	Default NVM Value	Description
15:0	RSA ExponentH	0x0	

6.3.80.4 Encrypted SHA256 Hash[n] (0x0082 + 1*n, n=0...127)

Bit(s)	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.3.81 Recovery FW Auth Header Section

Table 6-85. Recovery FW Auth Header Section Summary Table

Word Offset	Description	Section Reference
0x0000	Device Blank NVM Device ID	6.3.81.1
0x0001	Max Module AreaL	6.3.81.2
0x0002	Max Module AreaH	6.3.81.3
0x0003	Current Module AreaL	6.3.81.4
0x0004	Current Module AreaH	6.3.81.5
0x0005	Reserved	6.3.81.6
0x0006	Code Revision	6.3.81.7
0x0007	Reserved Spare Word	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	0x1888	

6.3.81.2 Max Module AreaL (0x0001)

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaL	0x1000	

6.3.81.3 Max Module AreaH (0x0002)

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaH	0x0009	

6.3.81.4 Current Module AreaL (0x0003)

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaL	0xE000	

6.3.81.5 Current Module AreaH (0x0004)

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaH	0x0006	

6.3.81.6 Reserved (0x0005)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

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 Recovery FW Image Section

This module is the whole module which has been signed by CSS.

It is in the NVM map for EM tool needs only.

6.3.83 DCB Rx Module Section

6.3.84 DCB Tx Module Section

6.3.85 Ext. CORER Registers Auto-Load Module Section

Default setup to registers and internal memories that load on CORER events.

Table 6-86. Ext. CORER Registers Auto-Load Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	ModuleLenL	6.3.85.1
0x0001	ModuleLenH	6.3.85.2

6.3.85.1 ModuleLenL (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ModuleLenL	0x00000000	Length in: 2 Bytes unit - 2 First Section -> Word: Ext. CORER Registers Auto-load Module -> ModuleLenL Last Section -> Word: Ext. CORER Registers Auto-load Module -> ModuleLenH

6.3.85.2 ModuleLenH (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ModuleLenH		

6.3.86 EMP Global Module Section

This section contains two sub-sections:

1. **Vendor-Specific Settings:** Settings for external PHY or Modules. This part has a proprietary format to enable advanced, vendor-specific, PHY setting. These settings are loaded into the external devices via the MDIO/I^2C interface.
2. **List of Qualified Modules:** Parameter list of up to 16 modules. Per module list holds OUI, Revision, and Version numbers.

Table 6-87. EMP Global Module Section Summary Table

Word Offset	Description	Section Reference
0x0000	Section Length	6.3.86.1
0x0001	Number of Qualified Modules	6.3.86.2
0x0002 + 12*n, n=0...15	Module OUI Bytes 0-1	6.3.86.3
0x0003 + 12*n, n=0...15	Module OUI Byte 2	6.3.86.4
0x0004 + 12*n, n=0...15	Vendor Part Number Bytes 0-1	6.3.86.5
0x0005 + 12*n, n=0...15	Vendor Part Number Bytes 2-3	6.3.86.6
0x0006 + 12*n, n=0...15	Vendor Part Number Bytes 4-5	6.3.86.7
0x0007 + 12*n, n=0...15	Vendor Part Number Bytes 6-7	6.3.86.8
0x0008 + 12*n, n=0...15	Vendor Part Number Bytes 8-9	6.3.86.9
0x0009 + 12*n, n=0...15	Vendor Part Number Bytes 10-11	6.3.86.10
0x000A + 12*n, n=0...15	Vendor Part Number Bytes 12-13	6.3.86.11
0x000B + 12*n, n=0...15	Vendor Part Number Bytes 14-15	6.3.86.12
0x000C + 12*n, n=0...15	Module Revision Number Bytes 0-1	6.3.86.13
0x000D + 12*n, n=0...15	Module Revision Number Bytes 2-3	6.3.86.14

6.3.86.1 Section Length (0x0000)

The length of the section in words. Note that section length does not include a count for the section length word.

Bit(s)	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

6.3.86.2 Number of Qualified Modules (0x0001)

Number of valid entries, 0 through 15, in the qualified modules' list.

Bit(s)	Field Name	Default NVM Value	Description
15:0	Number of Qualified Modules	0x10	

6.3.86.3 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].
- QSFP Vendor OUI is located at offsets [167:165].

Bit(s)	Field Name	Default NVM Value	Description
15:8	Byte 1	0x0	
7:0	Byte 0	0x0	

6.3.86.4 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].
- QSFP Vendor OUI is located at offsets [167:165].

Bit(s)	Field Name	Default NVM Value	Description
15:8	Reserved	0x0	Reserved.
7:0	Byte 2	0x0	

6.3.86.5 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
- QSFP+: Addr 183:168 page 0

Bit(s)	Field Name	Default NVM Value	Description
15:8	Byte 1	0x0	
7:0	Byte 0	0x0	

6.3.86.6 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
- QSFP+: Addr 183:168 page 0

Bit(s)	Field Name	Default NVM Value	Description
15:8	Byte 1	0x0	
7:0	Byte 0	0x0	

6.3.86.7 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
- QSFP+: Addr 183:168 page 0

Bit(s)	Field Name	Default NVM Value	Description
15:8	Byte 1	0x0	
7:0	Byte 0	0x0	

6.3.86.8 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
- QSFP+: Addr 183:168 page 0

Bit(s)	Field Name	Default NVM Value	Description
15:8	Byte 1	0x0	
7:0	Byte 0	0x0	

6.3.86.9 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
- QSFP+: Addr 183:168 page 0

Bit(s)	Field Name	Default NVM Value	Description
15:8	Byte 1	0x0	
7:0	Byte 0	0x0	

6.3.86.10 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
- QSFP+: Addr 183:168 page 0

Bit(s)	Field Name	Default NVM Value	Description
15:8	Byte 1	0x0	
7:0	Byte 0	0x0	

6.3.86.11 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
- QSFP+: Addr 183:168 page 0

Bit(s)	Field Name	Default NVM Value	Description
15:8	Byte 1	0x0	
7:0	Byte 0	0x0	

6.3.86.12 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
- QSFP+: Addr 183:168 page 0

Bit(s)	Field Name	Default NVM Value	Description
15:8	Byte 1	0x0	
7:0	Byte 0	0x0	

6.3.86.13 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
- QSFP+: Addr 185:184 page 0

Bit(s)	Field Name	Default NVM Value	Description
15:8	Byte 1	0x0	
7:0	Byte 0	0x0	

6.3.86.14 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
- QSFP+: Addr 185:184 page 0

Bit(s)	Field Name	Default NVM Value	Description
15:8	Byte 1	0x0	
7:0	Byte 0	0x0	

6.3.87 EMP Settings Module Header Section

This section contains the modes of operation of the EMP.

Table 6-88. EMP Settings Module Header Section Summary Table

Word Offset	Description	Section Reference
0x0000	Section Length	6.3.87.1
0x0001	Common Firmware Parameters	6.3.87.2
0x0002	FW Misc	6.3.87.3
0x0003	EEE Variables	6.3.87.4
0x0004	Maximal Wear-Out Value	6.3.87.5
0x0005	Initial Wear-Out Value	6.3.87.6
0x0006	Staggering Delay	6.3.87.7
0x0007	PXE PFC Timer Value	6.3.87.8
0x0008	PXE GPC High Threshold Value	6.3.87.9
0x0009	PXE GPC Low Threshold Value	6.3.87.10
0x000A	Internal Thermal Sensor Maximum Secured Value	6.3.87.11
0x000B	Internal Thermal Sensor Minimum Secured Value	6.3.87.12
0x000C	MAX_LL_AQ_CREDITS	6.3.87.13

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.

Bit(s)	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_LL_AQ_CREDITS

6.3.87.2 Common Firmware Parameters (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:10	Reserved	0x5	Reserved.
9	PF Reset on Queue Overflow	0b	0b = Disabled (default) 1b = Enabled
8:0	Reserved	0x3	Reserved.

6.3.87.3 FW Misc (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:1	FW Reserved	0x0	Reserved.
0	VEB Statistics Disable	0x0	0b = VEB Statistics are enable (default). 1b = VEB Statistics are disabled.

6.3.87.4 EEE Variables (0x0003)

Bit(s)	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 waits before it starts transmitting data after leaving the Low Power Idle (LPI) mode. This value is platform dependent and depends on the OEM platform.

6.3.87.5 Maximal Wear-Out Value (0x0004)

Bit(s)	Field Name	Default NVM Value	Description
15:0	MAX	0x1400	Maximal wear out credit value.

6.3.87.6 Initial Wear-Out Value (0x0005)

Bit(s)	Field Name	Default NVM Value	Description
15:0	INITIAL	0x800	Initial wear out credit value.

6.3.87.7 Staggering Delay (0x0006)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Delay	0x0	Port staggering delay in µs.

6.3.87.8 PXE PFC Timer Value (0x0007)

Bit(s)	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.

6.3.87.9 PXE GPC High Threshold Value (0x0008)

Bit(s)	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.

6.3.87.10 PXE GPC Low Threshold Value (0x0009)

Bit(s)	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.

6.3.87.11 Internal Thermal Sensor Maximum Secured Value (0x000A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Internal Thermal Sensor Maximum Secured Value	0x73	0-255 Recommended 0-115 degrees.

6.3.87.12 Internal Thermal Sensor Minimum Secured Value (0x000B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Internal Thermal Sensor Minimum Secured Value	0c60	0-255 Recommended 90-100 degrees.

6.3.87.13 MAX_LL_AQ_CREDITS (0x000C)

Bits	Field Name	Default NVM Value	Description
15:0	MAX_LL_AQ_CREDITS	0x8	

6.3.88 DL Scripts Section

6.3.89 Whitelist Section

6.3.90 Analog PHY Configuration Section

Includes static configurations of SerDes PHY registers for link establishment.

Table 6-89. Analog PHY Configuration Section Summary Table

Word Offset	Description	Section Reference
0x0000	Section Length Low	6.3.90.1
0x0001	Section Length High	6.3.90.2
0x0002	Reg Write Indirect List	6.3.90.3
0x0003	Reg Write Indirect List	6.3.90.4

6.3.90.1 Section Length Low (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Section Length Low		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

6.3.90.2 Section Length High (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Section Length High		

6.3.90.3 Reg Write Indirect List (0x0002)

Raw data module length: variable

6.3.90.4 Reg Write Indirect List (0x0003)

Raw data module length: variable

6.3.91 Configuration Metadata Section

NVM adaptive configuration metadata.

Table 6-90. Configuration Metadata Section Summary Table

Word Offset	Description	Section Reference

6.3.92 Control Pipe Package Section

This module contains the package used to configure the control pipe. This package is encoded as a set of Download Package Command (Opcode: 0x0C40) buffers.

Table 6-91. Control Pipe Package Section Summary Table

Word Offset	Description	Section Reference
0x0000	ModuleLenL	6.3.92.1
0x0001	ModuleLenH	6.3.92.2
0x0002	Package Raw	6.3.92.3

6.3.92.1 ModuleLenL (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ModuleLenL		Length in: 2 Bytes unit - 2 First Section -> Word: Control Pipe Package -> ModuleLenL Last Section -> Word: Control Pipe Package -> Package Raw

6.3.92.2 ModuleLenH (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:0	ModuleLenH	0x0	

6.3.92.3 Package Raw (0x0002)

Raw data module length: variable

6.3.93 EMP Image Section

This module contains the EMP processor code.

Table 6-92. EMP Image Section Summary Table

Word Offset	Description	Section Reference
0x0000	EMP Image Raw Data	6.3.93.1

6.3.93.1 EMP Image Raw Data (0x0000)

Raw data module length: variable

6.3.94 RDE Dictionaries Section

Table 6-93. RDE Dictionaries Section Summary Table

Word Offset	Description	Section Reference
0x0002	Dictionaries	6.3.94.1

6.3.94.1 Dictionaries (0x0000)

Raw data module length: variable

6.3.95 NVM Provisioning Area Section

Free area used as the new bank when updating 4 MB long modules.

Table 6-94. NVM Provisioning Area Section Summary Table

Word Offset	Description	Section Reference
0x21D000	Raw Data	6.3.95.1

6.3.95.1 Raw Data (0x21D000)

Raw data module length: variable

6.3.96 OROM Section

Option ROM module. It contains pre-boot code and settings read by BIOS.

Table 6-95. OROM Section Summary Table

Word Offset	Description	Section Reference
0x0000	OROM Data	6.3.96.1
0x0001	moduleTypeL	6.3.96.2
0x0002	moduleTypeH	6.3.96.3
0x0003	headerLenL	6.3.96.4
0x0004	headerLenH	6.3.96.5
0x0005	headerVersionL	6.3.96.6
0x0006	headerVersionH	6.3.96.7
0x0007	moduleIDL	6.3.96.8
0x0008	moduleIDH	6.3.96.9
0x0009	moduleVendorL	6.3.96.10
0x000A	moduleVendorH	6.3.96.11
0x000B	dateL	6.3.96.12
0x000C	dateH	6.3.96.13
0x000D	sizeL	6.3.96.14
0x000E	sizeH	6.3.96.15
0x000F	keySizeL	6.3.96.16
0x0010	keySizeH	6.3.96.17
0x0011	modulusSizeL	6.3.96.18
0x0012	modulusSizeH	6.3.96.19
0x0013	exponentSizeL	6.3.96.20
0x0014	exponentSizeH	6.3.96.21
0x0015	lad_srevL	6.3.96.22
0x0016	lad_srevH	6.3.96.23
0x0017 - 0x0018	Reserved	6.3.96.24
0x0019	lad_fw_entry_offsetL	6.3.96.25
0x001A	lad_fw_entry_offsetH	6.3.96.26
0x001B - 0x001C	Reserved	6.3.96.27
0x001D	lad_image_unique_idL	6.3.96.28
0x001E	lad_image_unique_idH	6.3.96.29
0x001F	lad_module_idL	6.3.96.30
0x0020	lad_module_idH	6.3.96.31
0x0021 - 0x0040	Reserved	6.3.96.32
0x0041 + 1*n, n=0...127	RSA Public Key	6.3.96.33
0x00C1	RSA ExponentL	6.3.96.34

Table 6-95. OROM Section Summary Table [continued]

Word Offset	Description	Section Reference
0x00C2	RSA ExponentH	6.3.96.35
0x00C3 + 1*n, n=0...127	Encrypted SHA256 Hash	6.3.96.36
0x0143	Device Blank NVM Device ID	6.3.96.37
0x0144	Max Module AreaL	6.3.96.38
0x0145	Max Module AreaH	6.3.96.39
0x0146	Current Module AreaL	6.3.96.40
0x0147	Current Module AreaH	6.3.96.41
0x0148	Reserved	6.3.96.42
0x0149	Code Revision	6.3.96.43
0x014A	Reserved Spare Word	6.3.96.44

6.3.96.1 OROM Data (0x0000)

Raw data module length: variable

6.3.96.2 moduleTypeL (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:0	moduleTypeL	0x6	

6.3.96.3 moduleTypeH (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:0	moduleTypeH		

6.3.96.4 headerLenL (0x0003)

Bit(s)	Field Name	Default NVM Value	Description
15:0	headerLenL	0xA1	

6.3.96.5 headerLenH (0x0004)

Bit(s)	Field Name	Default NVM Value	Description
15:0	headerLenH		

6.3.96.6 headerVersionL (0x0005)

Bit(s)	Field Name	Default NVM Value	Description
15:0	headerVersionL	0x00010000	

6.3.96.7 headerVersionH (0x0006)

Bit(s)	Field Name	Default NVM Value	Description
15:0	headerVersionH		

6.3.96.8 moduleIDL (0x0007)

Bit(s)	Field Name	Default NVM Value	Description
15:0	moduleIDL	0x0	

6.3.96.9 moduleIDH (0x0008)

Bit(s)	Field Name	Default NVM Value	Description
15	signMode	0x1	
14:0	moduleIDH		

6.3.96.10 moduleVendorL (0x0009)

Bit(s)	Field Name	Default NVM Value	Description
15:0	moduleVendorL	0x00008086	

6.3.96.11 moduleVendorH (0x000A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	moduleVendorH		

6.3.96.12 dateL (0x000B)

Bit(s)	Field Name	Default NVM Value	Description
15:0	DateL	0x20130530	0xMMDD

6.3.96.13 dateH (0x000C)

Bit(s)	Field Name	Default NVM Value	Description
15:0	DateH		0xYYYY

6.3.96.14 sizeL (0x000D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	sizeL	0x00025800	

6.3.96.15 sizeH (0x000E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	sizeH		

6.3.96.16 keySizeL (0x000F)

Bit(s)	Field Name	Default NVM Value	Description
15:0	keySizeL	0x40	

6.3.96.17 keySizeH (0x0010)

Bit(s)	Field Name	Default NVM Value	Description
15:0	keySizeH		

6.3.96.18 modulusSizeL (0x0011)

Bit(s)	Field Name	Default NVM Value	Description
15:0	modulusSizeL	0x40	

6.3.96.19 modulusSizeH (0x0012)

Bit(s)	Field Name	Default NVM Value	Description
15:0	modulusSizeH		

6.3.96.20 exponentSizeL (0x0013)

Bit(s)	Field Name	Default NVM Value	Description
15:0	exponentSizeL	0x1	

6.3.96.21 exponentSizeH (0x0014)

Bit(s)	Field Name	Default NVM Value	Description
15:0	exponentSizeH		

6.3.96.22 lad_srevL (0x0015)

Bit(s)	Field Name	Default NVM Value	Description
15:0	lad_srevL	0x0	

6.3.96.23 lad_srevH (0x0016)

Bit(s)	Field Name	Default NVM Value	Description
15:0	lad_srevH		

6.3.96.24 Reserved (0x0017 - 0x0018)

6.3.96.25 lad_fw_entry_offsetL (0x0019)

Bit(s)	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetL	0x14C	

6.3.96.26 lad_fw_entry_offsetH (0x001A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetH		

6.3.96.27 Reserved (0x001B - 0x001C)

6.3.96.28 lad_image_unique_idL (0x001D)

Bit(s)	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idL	0x0	

6.3.96.29 lad_image_unique_idH (0x001E)

Bit(s)	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idH		

6.3.96.30 lad_module_idL (0x001F)

Bit(s)	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

6.3.96.31 lad_module_idH (0x0020)

Bit(s)	Field Name	Default NVM Value	Description
15	Reserved	0b	Reserved.
14:0	lad_module_idH		

6.3.96.32 Reserved (0x0021 - 0x0040)

6.3.96.33 RSA Public Key[n] (0x0041 + 1*n, n=0...127)

Bit(s)	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.3.96.34 RSA ExponentL (0x00C1)

Bit(s)	Field Name	Default NVM Value	Description
15:0	RSA ExponentL	0x0	

6.3.96.35 RSA ExponentH (0x00C2)

Bit(s)	Field Name	Default NVM Value	Description
15:0	RSA ExponentH	0x0	

6.3.96.36 Encrypted SHA256 Hash[n] (0x00C3 + 1*n, n=0...127)

Bit(s)	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.3.96.37 Device Blank NVM Device ID (0x0143)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Device Blank NVM Device ID	0x1590	

6.3.96.38 Max Module AreaL (0x0144)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Max Module AreaL	0x1000	

6.3.96.39 Max Module AreaH (0x0145)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Max Module AreaH	0x0009	

6.3.96.40 Current Module AreaL (0x0146)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Current Module AreaL	0x1000	

6.3.96.41 Current Module AreaH (0x0147)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Current Module AreaH	0x0000	

6.3.96.42 Reserved (0x0148)

6.3.96.43 Code Revision (0x0149)

Bit(s)	Field Name	Default NVM Value	Description
15:8	Major Revision	0x0	
7:0	Minor Revision	0x0	

6.3.96.44 Reserved Spare Word (0x014A)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved Spare Word	0x0	

6.3.97 OROM Provisioning Area Section

600 KB OROM free area used as the new bank.

Table 6-96. OROM Provisioning Area Section Summary Table

Word Offset	Description	Section Reference
0x0000	Raw Data	6.3.97.1

6.3.97.1 Raw Data (0x0000)

Raw data module length: variable

6.3.98 Link Topology Netlist Section

MD link topology.

Table 6-97. Link Topology Netlist Section Summary Table

Word Offset	Description	Section Reference
0x0000	Link Topology Netlist Header	6.3.98.1
0x0001	Length	6.3.98.2
0x0002	Link Topology Netlist Data	6.3.98.3

6.3.98.1 Link Topology Netlist Header (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x11B	Valid values are: 0x11B = MD Link Topology

6.3.98.2 Length (0x0001)

Bit(s)	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

6.3.98.3 Link Topology Netlist Data (0x0002)

Raw data module length: 10236 words

Link topology netlist deliverable content.

6.3.99 TLV Extension Provisioning Area Section

Table 6-98. TLV Extension Provisioning Area Section Summary Table

Word Offset	Description	Section Reference
0x0000	Sub Module Type - Padding	6.3.99.1
0x0001	Length	6.3.99.2

6.3.99.1 Sub Module Type - Padding (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0xFFFF	Valid values are: 0xffff = Padding Module

6.3.99.2 Length (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Length		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

6.3.100 Link Topology Scratch Pad Area Section

Link topology scratch pad area used by firmware only.

Table 6-99. Link Topology Scratch Pad Area Section Summary Table

Word Offset	Description	Section Reference
0x0000	[New Word]	6.3.100.1

6.3.100.1 [New Word] (0x0000)

Raw data module length: variable

6.3.101 FW Scratch Pad Area Section

Firmware scratch pad area used by firmware only.

Table 6-100. FW Scratch Pad Area Section Summary Table

Word Offset	Description	Section Reference
0x0000	[New Word]	6.3.101.1

6.3.101.1 [New Word] (0x0000)

Raw data module length: variable

6.3.102 Factory Settings Header Section

Table 6-101. Factory Settings Header Section Summary Table

Word Offset	Description	Section Reference
0x0000	Actual Size L	6.3.102.1
0x0001	Actual Size H	6.3.102.2
0x0002	Password	6.3.102.3
0x0003	TLV Extension Offset	6.3.102.4
0x0004	TLV Extension Size	6.3.102.5
0x0005	PCIR AL Offset	6.3.102.6
0x0006	POR AL Offset	6.3.102.7
0x0007	PCI Serial Id MAC Address Offset	6.3.102.8
0x0008 - 0x000F	Reserved	6.3.102.9

6.3.102.1 Actual Size L (0x0000)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Actual Size L	0xFFFF	

6.3.102.2 Actual Size H (0x0001)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Actual Size H	0xFFFF	

6.3.102.3 Password (0x0002)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Password	0xFFFF	

6.3.102.4 TLV Extension Offset (0x0003)

Bit(s)	Field Name	Default NVM Value	Description
15:0	TLV Extension Offset	0xFFFF	

6.3.102.5 TLV Extension Size (0x0004)

Bit(s)	Field Name	Default NVM Value	Description
15:0	TLV Extension Size	0xFFFF	

6.3.102.6 PCIR AL Offset (0x0005)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PCIR AL Offset	0xFFFF	

6.3.102.7 POR AL Offset (0x0006)

Bit(s)	Field Name	Default NVM Value	Description
15:0	POR AL Offset	0xFFFF	

6.3.102.8 PCI Serial ID MAC Address Offset (0x0007)

Bit(s)	Field Name	Default NVM Value	Description
15:0	PCI Serial ID MAC Address Section	0xFFFF	

6.3.102.9 Reserved (0x0008 - 0x000F)

6.3.103 Factory Settings Area Section

Table 6-102. Factory Settings Area Section Summary Table

Word Offset	Description	Section Reference
0x0000 + 1*n, n=0...26607	Reserved.	6.3.103.1

6.3.103.1 Reserved[n] (0x0000 + 1*n, n=0...26607)

Bit(s)	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved.

6.3.104 Guarded Zone Section

Guarded zone for firmware use. The zone is protected such that only firmware can read and write to it.

Table 6-103. Guarded Zone Section Summary Table

Word Offset	Description	Section Reference
0x0000	[New Word]	6.3.104.1

6.3.104.1 [New Word] (0x0000)

Raw data module length: variable

Chapter 7 Packet Processing

7.1 Introduction

The E810 provides a programmable packet processing engine with extensive support for advanced applications ranging from Enterprise LAN to SDN/NFV, Cloud, Comms, and Wireless.

Key features include:

- FlexiPipe virtualized pipeline with profile-based processing stages.
- OpenFlow-enabled Flow-Tables structure with metadata propagation between pipeline stages.
- P4-ready parsing and classification flows.
- Multiple binary and ternary (like ACL) classification stages.
- RDMA Flow Table expansions to host DRAM.
- SDN-enabled architecture.
  - Generic L1 to L5+ protocol handling.
  - Protocol-agnostic SDN parsing, classification, and packet forwarding and modification, including standard L2 and/or L3+ packet switching.
  - N-Tuple lookup key classification.
  - M-Tuple action execution.
- Deep header processing up to 504 bytes.
  - Supports tunneled overlay encapsulations and large IPv6 extension headers.
- Fully software-programmable FlexiParser.
  - Supports user-defined proprietary packet formats.
  - Programmable parse graph, enforcing user-defined parsing policy.
  - Programmable PTYPEs (Packet Types).
- Cloud-enabled generic overlays including GRE, NVGRE, VXLAN, VXLAN-GPE, Geneve.
- MPLS-enabled processing.
  - Outer MPLS and inner-MPLS (e.g., MPLSoGRE, MPLSoUDP).
- Service chaining forwarding including NSH (Network Service Header) processing.

The advanced packet processing engine enables high-performance acceleration by hardware-offloading of software functions, especially in a VMM/Hypervisor environments with SR-IOV paths.

7.2 FlexiPipe Processing Pipeline

The E810 FlexiPipe pipeline serves as the backbone of the packet processing engine. The OpenFlow-enabled pipeline is fully SDN/NFV-enabled, providing multiple programmable lookup stages (Flow Tables) augmented with programmable FlexiParser and packet modifier.

The generic FlexiPipe pipeline architecture enables support for diverse processing requirements. The pipeline provides multiple protocol-agnostic processing stages that can be re-purposed according to the target application. For example, a generic classification stage in the pipe can set a Destination VSI/ Destination Q in one application, whereas in another application the same generic classification stage can set an opaque metadata associated with the packet. Each of these classifications can be performed using different lookup keys over different protocols.

7.2.1 Rx and Tx Pipeline Structure

The E810 pipeline shares sections of the pipe between Rx and Tx flows. Shared components are designed to process packets at the combined rates of Rx and Tx.

The pipeline provides a loopback forwarding path between Tx and Rx for packets that are destined to functions residing on the same host. Such packets are first processed in the Tx section of the pipeline, stored in the Tx Packet Buffer, copied to Rx Packet Buffer, and then processed in the Rx section of the pipeline.

The Rx processing pipeline contains the following stages for LAN traffic:

1. Rx Packet Buffer
2. Rx FlexiParser (Section 7.7)
3. Switch (Binary Classifier) — Default function: VEB switch, multi-cast and mirror replicate (Section 7.8)
4. ACL (Ternary Classifier) — Default function: ACL (Section 7.9)
5. Classifications filters (Section 7.10)
   - FD (Binary Classifier) — Default function: Flow Director (Q steering)
   - QH (Binary Classifier) — Default function: RDMA flow steering
   - Hash — Default function: RSS
6. Flex Descriptor Builder (Section 7.6)
7. Packet Modifier

A detailed view of the FlexiPipe Rx processing pipeline is illustrated in Figure 7-1.

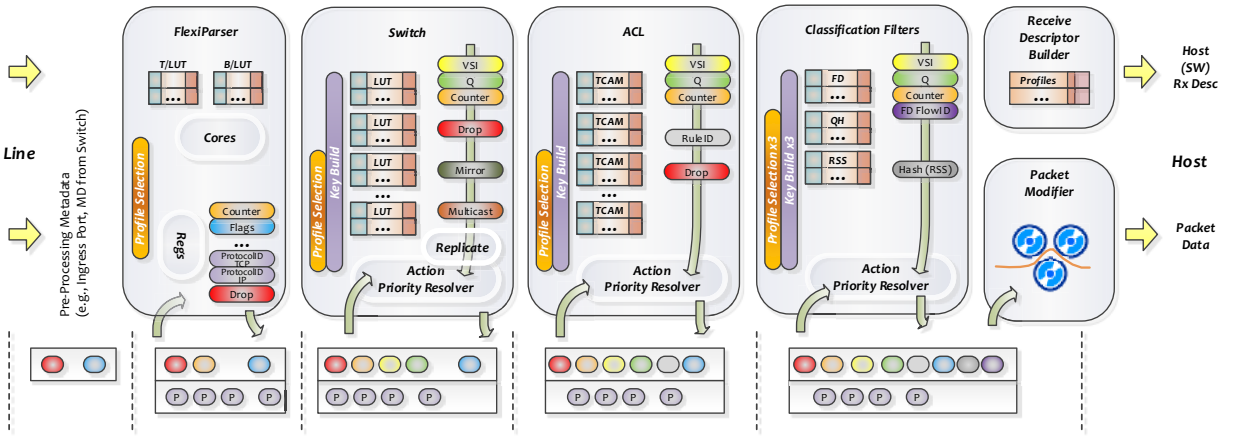


Figure 7-1. Rx Processing Pipeline (Showing Post-Receive Packet Buffer Section)

The FlexiPipe Tx processing pipeline contains the following stages:

1. Tx WFQ Scheduler (Section 8.3)
2. LSO/TSO Segmentation Offload (Section 10.5.8.4)
3. Packet Modifier
4. Tx FlexiParser (Section 7.7)
5. Switch (Binary Classifier) - Default function: Packet steering to LAN (Line) / LB (Loopback to host) (Section 7.8)
6. ACL (Ternary Classifier) - Default function: ACL (Section 7.9)
7. Tx Packet Buffer

A detailed view of the FlexiPipe Tx processing pipeline is illustrated in Figure 7-2.

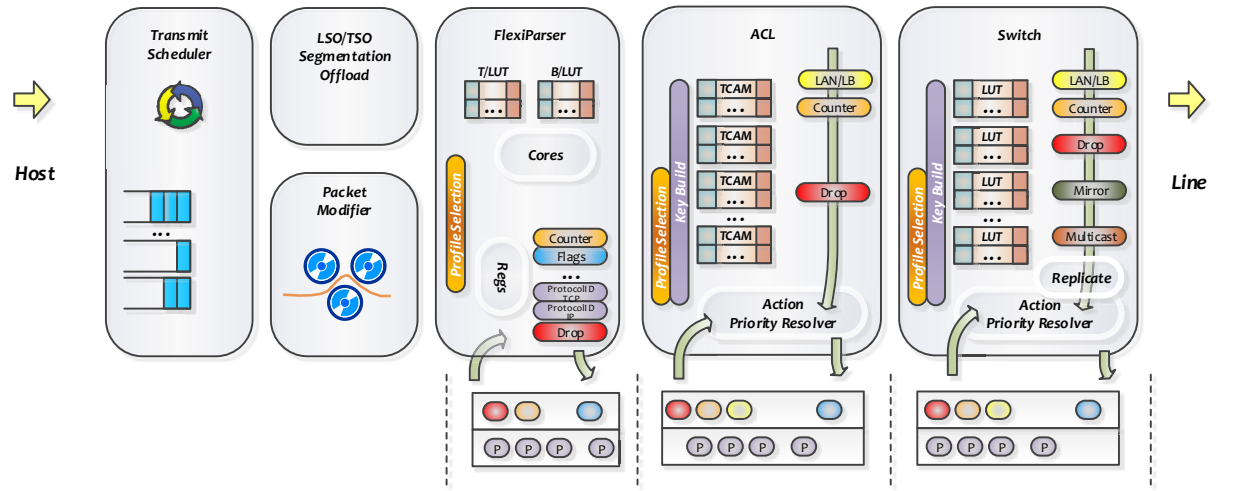


Figure 7-2. Tx Processing Pipeline (Showing Pre-Transmit Packet Buffer Section)

Further descriptions of the pipeline stages appear in subsequent sections of this document.

7.2.2 Pipeline Virtualization

The E810 FlexiPipe provides a unique feature of Pipeline Virtualization that logically partitions the single physical pipeline instance into functionally-independent logical pipeline instances. Each behaves like a standalone pipeline available for programming, as illustrated in Figure 7-3.

In Figure 7-3, the physical pipeline is virtualized and partitioned into two logical pipeline instances: orange and green. Each of the logical pipeline instances can be programmed independently and provide different functionality.

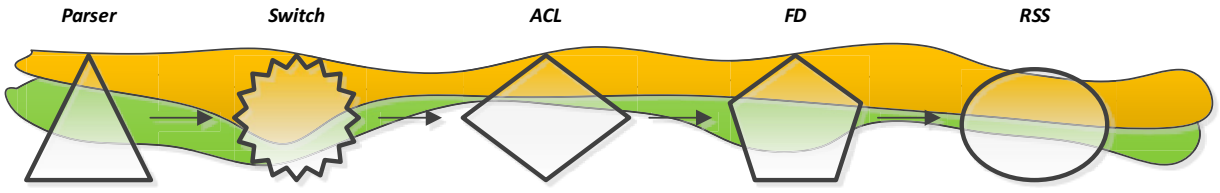


Figure 7-3. Pipeline Virtualization

7.3 Priority Resolver

The E810 pipeline provides a generic action-agnostic metadata generation at each pipeline stage. Consequently, multiple pipeline stages are allowed to generate the same MDID types, possibly creating conflicts.

For example, multiple pipeline stages might attempt to set the <Destination Q> of the packet: the switch, the ACL, and the FD (Flow Director). Similarly, multiple pipeline stages might attempt to set the <DestinationVSI>.

To guarantee consistent behavior, the pipeline provides a metadata Priority Resolver at each pipeline stage, resolving possible conflicts per each MDID type. The Priority Resolver selects the highest priority metadata value per each MDID metadata type, based on the input metadata to the pipeline stage and the metadata generated by the pipeline stage itself (the core). When priorities are equal, the later pipeline stage wins (for example, if switch selects <DestinationVSI>=A Priority=3, and ACL selects <DestinationVSI>=B Priority=3, the ACL wins because it is a later stage in the pipe).

The metadata Priority Resolver is illustrated in Figure 7-4.

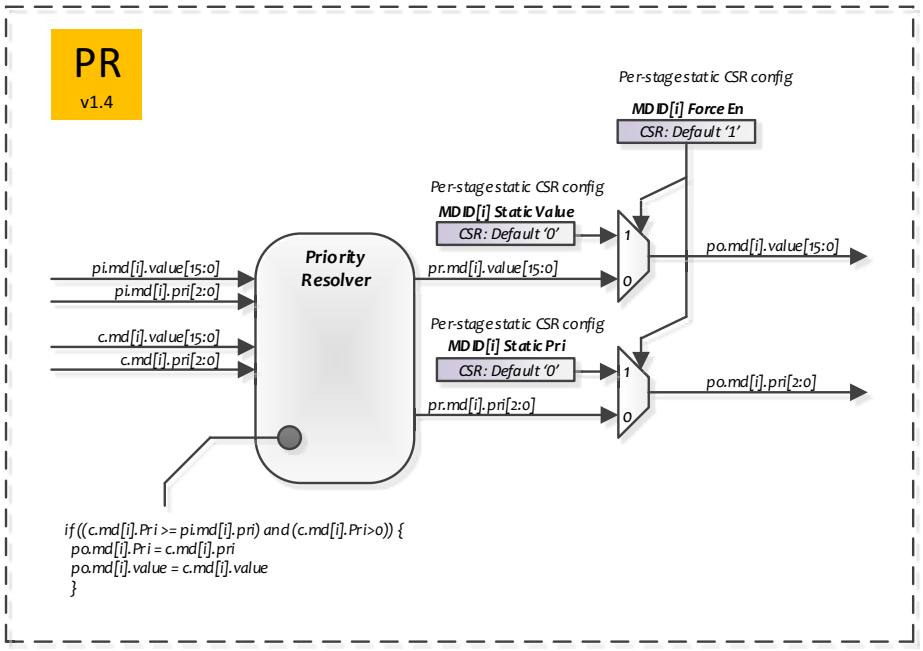


Figure 7-4. Metadata Priority Resolver

MDID metadata types that have priority association are carried in the pipeline alongside with their priority association and provided to the next pipeline stage for further processing.

7.3.1 MDID Override

The Priority Resolver in each pipeline stage provides an MDID Override feature for setting the MDID outputs of the pipe stage from CSRs. For each RW MDID metadata, the following CSRs are provided:

- <MDID[i] Force Enable> — When set, MDID override is enabled for MDID type i.
- <MDID[i] Static Value> — When MDID override is enabled for MDID type i, the metadata value is taken from CSR.
- <MDID[i] Static Priority> — When MDID override is enabled for MDID type i, the metadata priority is taken from CSR.

The metadata override feature enables functional bypassing of pipeline stages, complete or partial bypass, depending on the target use case. It also facilitates the silicon bring-up process by placing the pipeline into a simple static configuration where all traffic is directed to known destinations (VSI, Q, and so on) determined by CSRs.

7.3.2 Programming the Priority Resolver

As detailed in earlier sections, the E810 FlexiPipe implements priority resolution mechanisms for several of its actions.

These mechanisms are placed in each of the four stages of the Routing Control Unit (RCU): the parser, the switch, the ACL, and the Rx filters.

7.4 FlexActions

When a pipe stage generates FlexActions, they are mapped into the MDID metadata infrastructure of the pipeline. A generic FlexAction structure is made of xM[1] FlexAction Tuples that define the actions generated and the mapping of these actions into the pipeline metadata.

Note: Some legacy blocks in the E810 pipeline provide a hybrid FlexAction implementation, where some actions are statically mapped (for example, VSI selection in the legacy portion of the E810 switch), while the block also provides a FlexAction structure for generating all the other actions and metadata supported by the pipeline (like the flex portion of the E810 switch). More details are provided in further sections of this document that refer to the corresponding pipeline stages.

An example for the generic structure of a FlexAction is illustrated in Figure 7-5.

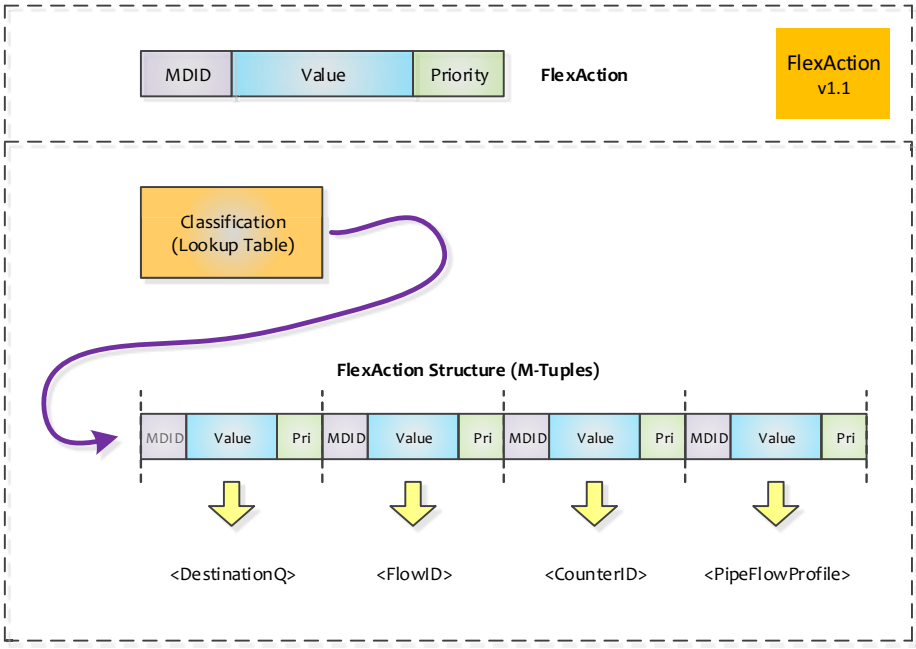


Figure 7-5. FlexAction Structure

The FlexAction structure is made of xN FlexAction Tuples, each carrying the following fields:

- <MDID> — The MDID type to be assigned, such as <DestinationQ>.
- <Value> — The value to associate with the MDID, such as the <DestinationQ> number.
- <Priority> — The priority to associate with the Action Tuple, for resolving conflicts with other sources in the pipe that can generate the same MDID type.

Note: <Priority>=0 indicates that the MDID is invalid and its <Value> should be disregarded.

1. xM is an implementation parameter selected independently per each pipeline function that generates actions. It denotes the number of FlexActions generated simultaneously, e.g., as a result of a classification lookup hit. In the example in Figure 7-5, M=4 and the FlexActions generated are <DestinationQ>, <FlowID>, <CounterID>, and a user-defined <PipeFlowProfile>.

7.5 Extractor

The E810 pipeline provides a generic extraction logic instantiated at pipeline stages that require extraction from the packet header, namely:

- Switch
- ACL
- Rx filters: FD (Flow Director), HASH, RDMA/QH (Quad Hash)

The Extractor's logic uses the Profile ID, generated by the Profile Builder, and the metadata supplied by previous stages in the pipeline to extract fields to be used in each stage.

Each extractor contains a configuration per Profile ID, which determines which fields from the packet header or its metadata are used for packets that hit that Profile ID. The extractor output, which contains the set of fields selected for the packet, is referred as the *Fields Vector*.

The extractor offsets are expressed in byte resolution while the extraction itself is done in word resolution. For each word in the Fields Vector, the configuration must specify a source word that can be from either the packet's header (up to 504B first bytes) or the packet metadata. Furthermore, a source offset in the packet's header is expressed as a relative offset to a protocol offset, using the protocol IDs offset (Section A.4), thus leveraging the capabilities of the programmable parser.

Note: Any configuration that causes the extractor to cross the header length boundaries (either by crossing the maximal supported header size or by crossing the packet length) is illegal, but an extraction can cross boundaries of internal protocols and can also refer to parts of the payload that are within the extracted header.

7.5.1 Programming the Extraction Logic

In general, the extraction logic must be programmed after the relevant profile builder is configured and the Profile ID is allocated. This section describes the configuration method used to program the extraction sequence for a previously allocated Profile ID. When an incoming packet is associated with that Profile ID, the extractor logic must use this configuration to extract fields from the packet's header and metadata to create a fields vector that are used in the given stage.

Important: The E810 pipeline supports extraction from packet headers of up to 504 byes (that is, only from the 504 bytes trailing the start-of-frame delimiter) or the length of the packet (the lowest of both). Any extractor configuration that relates to an offset beyond the maximal packet size or beyond the maximal supported header size is invalid and can cause unexpected hardware behavior.

The extractor CSRs are listed in Table 7-1. Each CSR mentioned in Table 7-1 is instantiated in three pipeline stages (Switch, ACL, and Rx filters), though some might be valid only in specific stages. Therefore, the three rightmost columns in the table specify which of the CSRs is valid in each stage and its functionality in the relevant stage.

Furthermore, each extractor in each stage supports a different output width, based on the needs of that stage. Table 7-2 specifies the output size of each extractor in each stage.

Important: Attempting to configure an extractor to output more words than it supports is forbidden and can cause unexpected hardware behavior.

Table 7-1. Extractor CSRs

CSR	Description	Switch	ACL	Rx Filters
PRFLM_DATA_0	Each CSR is used to configure protID (see Section A.4)/offset of extraction word N (See Table 7-2 for applicable FV sizes in each stage) for FV of sub-extractor #i (i=[0..2]) of this extractor. If the specified protocol ID is smaller than 255, the source is taken from packet using the protocol ID's offset plus the offset in the protocol (in bytes). If the specified offset is equal to 255, the source is taken from the packet's metadata according to the following rules: • Offset 0-30: The word is extracted from the pipe status bus. The source word byte offset is the value specified in this field. • Offset 32-62: The word is extracted from the packet status bus. The source word byte offset is the value specified in this field minus 32 (for example, a value of 32 extracts a word from byte offset 0 in the packet status bus). ***Note:*** The byte offset values mentioned above should not be confused with the MDID. ***Note:*** Offset values 31 and 63 are invalid, as word boundary crosses the relevant status bus width.	Main switch	ACL	Hash
PRFLM_DATA_1		MNG	Invalid	FD
PRFLM_DATA_2		Invalid	Invalid	Quad-Hash
PRFLM_CTRL[0]	When the CSRs is written, it determines the profileID within sub-extractor #i (i=0..2) for which the FV extractions determined in PRFLM_data_i are configured. This CSR is to be written only after all relevant PRFLM_data_i have the correct value. This CSR is also used to read the relevant content into the relevant PRFLM_data_i array. RD/WR is controlled by a field in this CSR.	Main switch	ACL	Hash
PRFLM_CTRL[1]		MNG	Invalid	FD
PRFLM_CTRL[2]		Invalid	Invalid	Quad-Hash

Table 7-2. Extractor Output Width per Stage

Stage	Extractor Instance	Output Width (Words)
Switch	Main switch	48
	Manageability	32
ACL	ACL	32
Rx Filters	Hash	24
	Quad-hash	32 (24 + 8 for fixed fields)
	Flow-director	24

7.5.1.1 Programming Flow

When programming an extraction sequence for a Profile ID #n, the programming entity follows the flow below:

1. For each of the words in the fields vector, define the location of the source word that will be placed in the fields vector:
   a. Set PRFLM_DATA_i.*PROT* to the protocol ID whose offset in the packet's header will be used as a base offset for the source word.
   b. Set PRFLM_DATA_i.*OFF* to the byte offset relative to the protocol ID offset. This field's value is added to the protocol offset to generate the final offset of the source word.

Note: When the PRFLM_DATA_i.*PROT* field is set to 255, the source word is taken from the packet's metadata and not from the packet's header. In this case, the value of the PRFLM_DATA_i.*OFF* field determines the source in the packet's metadata as described in Table 7-1.

2. Once the sources for all words in the fields vector were configured, complete the programming:
   a. Set PRFLM_CTRL[i].*PRFL_IDX* to the Profile ID for which the extraction sequence is being programmed.
   b. Set PRFLM_CTRL[i].*WR_REQ*.
3. Pull for the value of PRFLM_CTRL[i].WR_REQ. Once it is cleared, the programming is completed.

7.5.1.2 Querying Flow

The extractor logic supports reading back an extraction sequence for a given Profile ID. This flow can be useful for implementing a read-modify-write mechanism allowing a partial change in configuration.

1. Trigger the read of the configuration:
   a. Set PRFLM_CTRL[i].*PRFL_IDX* to the Profile ID for which the extraction sequence is being read.
   b. Set PRFLM_CTRL[i].*RD_REQ*.
2. Pull for the value of PRFLM_CTRL[i].*RD_REQ*. Once it is cleared, the read is completed.
3. The values are available in the PRFLM_DATA_i array.

7.6 Receive Descriptor Builder

7.6.1 Overview

The E810 provides new programmable receive descriptor formats on top of the legacy (backward compatible) receive descriptor formats. The new Flex Descriptor feature virtualizes the hardware/ software interface on the receive path, allowing software entities to program customized receive descriptors for specific use cases and applications. The feature allows each PF/VF/VM to customize its own Flex Descriptor structure with different fields and flags, exposing to software pre-processing results of interest such as <FlowID>, <ACLRuleID>, offsets to protocol headers or other opaque metadata that generated by the E810.

The Receive Descriptor Builder stage in the E810 pipeline provides the programming infrastructure and the composition logic of the Flex Descriptors.

The generic structure of the receive Flex Descriptor is illustrated in Figure 7-6.

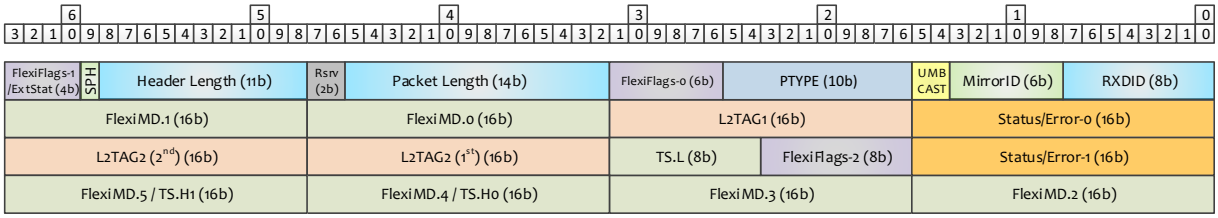


Figure 7-6. Receive Flex Descriptor

Note: Figure 7-6 shows the structure of the 32-byte receive Flex Descriptor. The shorter 16-byte receive Flex Descriptor uses the first 16 bytes of this structure, while the last 16-byte chunk of the structure is omitted.

The E810 provides 63 x Descriptor Builder Profiles for programming different descriptor formats, including the 16-byte and 32-byte legacy descriptor formats. The *RXDID* field in the Flex Descriptor structure identifies the profile and hence the descriptor format delivered to software, in particular the contents of fields and flags.

Note: *RXDID*=7 is reserved and should not be used (in legacy descriptors, *MIRR*=7 means "mirror packet", and that is the only way to distinguish it from Flex Descriptors).

The Descriptor Builder Profile selected for each packet is derived from two sources:

- **Pipeline Actions** — Based on rules programmed into the pipeline (for example, if the packet matches a rule in the ACL, the action of that rule might request a descriptor profile that exposes the <ACLRuleID> via the Receive Flex Descriptor).
- **Queue Context** — Based on the Destination Q of the packet (for example, packets sent to specific destination queue can have the same descriptor format, like the 16-byte legacy format).

The profile request is an MDID metadata/action carried in the pipeline. Conflicting actions from multiple sources that request different profiles for the same packet are resolved through the standard MDID priority resolution scheme described in Section 7.3.

7.6.2 Legacy Descriptor Format

The E810 preserves the legacy (backward compatible) descriptor format (see Section 10.4.2).

The legacy formats are available for selection as Descriptor Builder Profiles #0 (16-byte) and #1 (32-byte):

- **Descriptor Builder Profiles #0** — Generates the legacy 16-byte descriptor format, identified by *RXDID*=0 in the descriptor structure delivered to software.
- **Descriptor Builder Profiles #1** — Generates the legacy 32-byte descriptor format, identified by *RXDID*=1 in the descriptor structure delivered to software.

Note: *RXDID*=7 is reserved and should not be used (in legacy descriptors, *MIRR*=7 means "mirror packet", and that is the only way to distinguish it from flex descriptors).

The legacy descriptor can be used in applications where it is required to preserve backward compatibility at the driver level. For example, using the legacy X710/XXV710/XL710 VF driver in VFs running over the E810.

The structure of the 32-byte legacy (backward compatible) descriptor is illustrated in Figure 7-7.

The first two QW (Quad Words = 16B) are identical to the 16B legacy descriptor format.

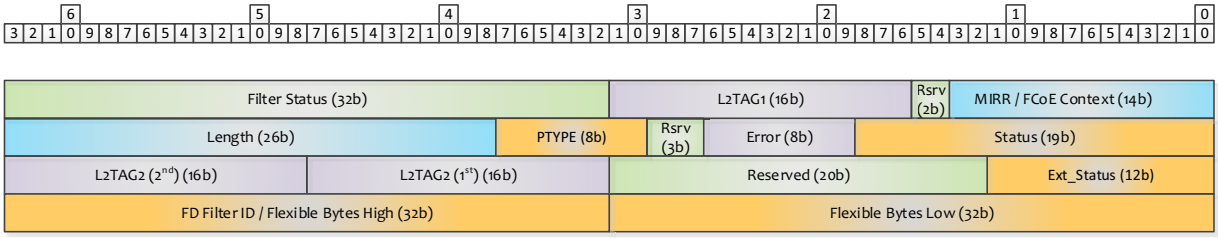


Figure 7-7. Receive Descriptor (32-Byte) - Legacy Format

7.6.2.1 PTYPE Translation

The legacy (backward compatible) descriptor uses an 8-bit PTYPE, whereas the E810 pipeline provides an advanced 10-bit PTYPE. The E810 provides a PTYPE Translation Table for matching the 10-bit E810 PTYPEs to the legacy 8-bit PTYPEs.

Note: The PTYPE Translation Table also provides the legacy 8-bit PTYPE to the Receive Data Processing Unit (RDPU) for the purpose of checksum generation.

The factory parsing program loaded from NVM provides the default 10-bit PTYPEs and their translations to legacy 8-bit PTYPEs. Further information is found in Appendix A.

7.6.3 Flex Descriptor Format

The Flex Descriptor includes permanent infrastructure fields that appear in all Flex Descriptor formats and user-defined fields that can be programmed differently by different Descriptor Builder Profiles:

- Permanent infrastructure fields:
  - RXDID
  - MirrorID
  - UMBCAST
  - PTYPE
  - Packet Length
  - Header Length
  - SPH
  - Status/Error field(s)
  - L2TAG fields
- User-defined fields:
  - FlexiFlags (user-selectable indications)
  - FlexiMD (user-selectable Metadata) and TS (Timestamp)

Detailed structure of the Flex Descriptor is illustrated in Figure 7-8.

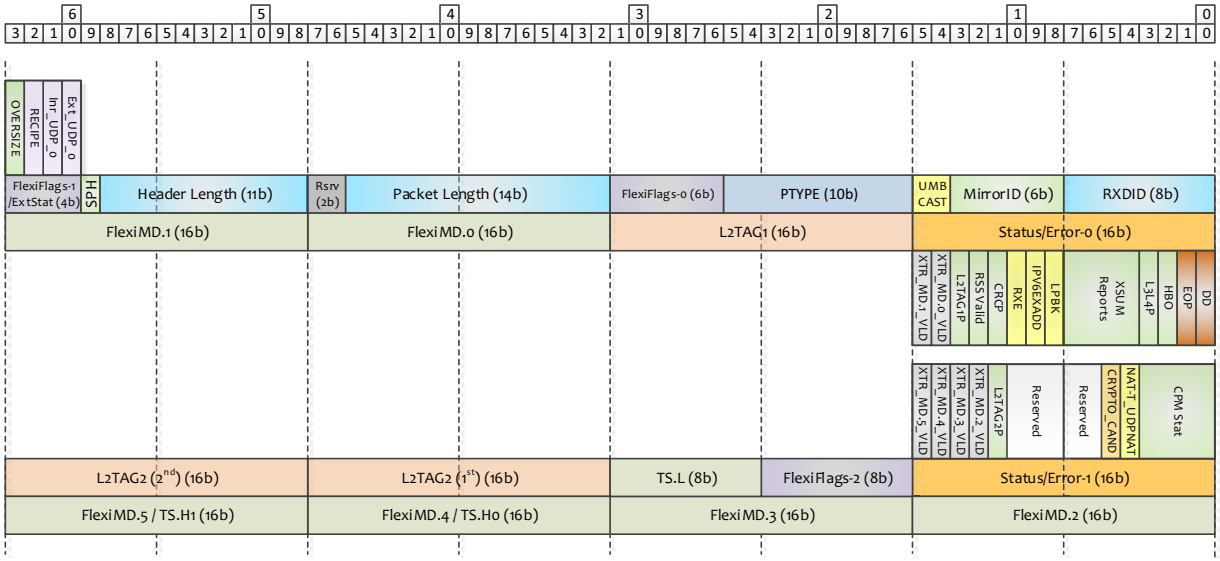


Figure 7-8. Receive Flex Descriptor (with Flags)

Table 7-3. Receive Flex Descriptor Fields

Word	Field Mnemonic	Width (bits)	Description
16-byte/32-byte Receive Flex Descriptors			
0	RXDID	8	**Descriptor Builder ProfileID** — Denoted the number of profile entry selected for this descriptor. Profiles #0 - Generates the legacy 16B descriptor format, identified by *RXDID*=0. Profiles #1 - Generates the legacy 32B descriptor format, identified by *RXDID*=1.
	MirrorID	6	**Mirror ID** — Identical to the field definition in the legacy descriptor format.
	UMBCAST	2	**UMBCAST** — Identical to the field definition in the legacy descriptor format.
1	PTYPE	10	**Packet Type** — Enumerates the packet type according to the packet types programmed into the device. The Flex Descriptor *PTYPE* is a 10-bit field. It can carry different packet type encoding than the legacy 8-bit *PTYPE* field in legacy descriptors.
	FlexiFlags.0	6	**Flexible Flags Section 0** — Up to 6 x user-selectable flags per *RXDID ProfileID*.
2	Packet Length	14	**Packet Length** — Identical to the field definition in the legacy descriptor format.
	Reserved	2	Reserved.
3	Header Length	11	**Header Length** — Identical to the field definition in the legacy descriptor format.
	SPH	1	**SPH** — Identical to the field definition in the legacy descriptor format.
	FlexiFlags.1/ExtStats	4	**Flexible Flags Section 1/Extended Status** — Up to 4 x user-selectable flags per *RXDID ProfileID*. This field can also carry additional status information: Ext_UDP_0, Int_UDP_0, RECIPE, OVERSIZE.
4	Status/Error.0	16	**Status/Error Section 0** — Status and error indications.
5	L2TAG1	16	**L2TAG1** — Identical to the field definition in the legacy descriptor format.
6	FlexiMD.0	16	**Flexible Metadata Container #0** — Generic 16-bit container for conveying metadata/ actions from the E810 pipeline to the host.
7	FlexiMD.1	16	**Flexible Metadata Container #1** — Generic 16-bit container for conveying metadata/ actions from the E810 pipeline to the host.

Table 7-3. Receive Flex Descriptor Fields [continued]

Word	Field Mnemonic	Width (bits)	Description
32-byte Receive Flex Descriptors Only			
8	Status/Error.1	16	**Status/Error Section 1** — Additional status and error indications.
9	FlexiFlags.2	8	**Flexible Flags Section 2** — Up to 8 x user-selectable flags per *RXDID ProfileID*.
	TS.L	8	**TimeStamp Word Low** — Lower word of the TimeStamp value.
10	L2TAG2 (1st)	16	**L2TAG2 (1st)** — Identical to the field definition in the legacy descriptor format.
11	L2TAG2 (2nd)	16	**L2TAG2 (2nd)** — Identical to the field definition in the legacy descriptor format.
12	FlexiMD.2	16	**Flexible Metadata Container #2** — Generic 16-bit container for conveying metadata/ actions from the E810 pipeline to the host.
13	FlexiMD.3	16	**Flexible Metadata Container #3** — Generic 16-bit container for conveying metadata/ actions from the E810 pipeline to the host.
14	FlexiMD.4/TS.H0	16	**Flexible Metadata Container #4/TimeStamp Word High #0** — Generic 16-bit container for conveying metadata/actions from the E810 pipeline to the host / Higher word #0 of the timestamp value.
15	FlexiMD.5/TS.H1	16	**Flexible Metadata Container #5/TimeStamp Word High #1** — Generic 16-bit container for conveying metadata/actions from the E810 pipeline to the host / Higher word #0 of the timestamp value.

7.6.3.1 Status/Error.0 Field

Table 7-4 details the contents of the *Status/Error.0* field.

Table 7-4. Status/Error.0 Field (16b)

Bit	Field Mnemonic	Width (Bits)	Description
0	DD	1	**Descriptor Done** — Identical to the filed definition in the legacy descriptor format.
1	EOP	1	**End Of Packet** — Identical to the filed definition in the legacy descriptor format.
2	HBO	1	**Header Buffer Overflow** — Identical to the filed definition in the legacy descriptor format.
3	L3L4P	1	**L3 and L4 Integrity Check** — Identical to the field definition in the legacy descriptor format.
7:4	XSUM Reports	4	**Checksum Reports** — Checksum error indications: bit 4 = IPE — IP checksum error indication (for tunneled packets it is the most inner IP header indication). bit 5 = L4E — L4 integrity error indication (most inner L4 header in case of UDP tunneling). bit 6 = EIPE — External (most outer) IP header (only relevant for tunneled packets). bit 7 = EUDPE — External (most outer) UDP checksum error (only relevant for tunneled packets).
8	LPBK	1	**Loopback** — Identical to the field definition in the legacy descriptor format.
9	IPV6EXADD	1	**IPv6 with Destination Options Header or Routing Header** — Identical to the field definition in the legacy descriptor format.
10	RXE	1	**Receive MAC Errors** — Identical to the field definition in the legacy descriptor format.
11	CRCP	1	**Ethernet CRC Present** — Identical to the field definition in the legacy descriptor format.
12	RSS/HASH Valid	1	**RSS/HASH Valid** — Indicates that the RSS/HASH result is valid. Qualifies RSS/HASH metadata in *FlexiMD* fields.

Table 7-4. Status/Error.0 Field (16b) [continued]

Bit	Field Mnemonic	Width (Bits)	Description
13	L2TAG1P	1	**L2 Tag 1 Presence** — Identical to the field definition in the legacy descriptor format.
14	XTR_MD.0_VLD	1	**Extract MD.0 Valid** — Indicates that extracted data from the packet is valid in MD.0. Extraction is performed according to the *ProfileID* entry in *RXDID*.
15	XTR_MD.1_VLD	1	**Extract MD.1 Valid** — Indicates that extracted data from the packet is valid in MD.1. Extraction is performed according to the *ProfileID* entry in *RXDID*.

7.6.3.2 Status/Error.1 Field

Table 7-5 details the contents of the *Status/Error.1* field.

Table 7-5. Status/Error.1 Field (16b)

Bit	Field Mnemonic	Width (Bits)	Description
0-3	Reserved	4	Reserved.
4	NAT-T/UDP-NAT	1	**NAT-T / UDP-NAT** — The packet is a UDP tunneled packet (including UDP-NAT-ESP).
10:5	Reserved	6	Reserved.
11	L2TAG2P	1	**L2 Tag 2 Presence** — Identical to the field definition in the legacy descriptor format.
12	XTR_MD.2_VLD	1	**Extract MD.2 Valid** — Indicates that extracted data from the packet is valid in MD.2. Extraction is performed according to the *ProfileID* entry in *RXDID*.
13	XTR_MD.3_VLD	1	**Extract MD.3 Valid** — Indicates that extracted data from the packet is valid in MD.3. Extraction is performed according to the *ProfileID* entry in *RXDID*.
14	XTR_MD.4_VLD	1	**Extract MD.4 Valid** — Indicates that extracted data from the packet is valid in MD.4. Extraction is performed according to the *ProfileID* entry in *RXDID*.
15	XTR_MD.5_VLD	1	**Extract MD.5 Valid** — Indicates that extracted data from the packet is valid in MD.5. Extraction is performed according to the *ProfileID* entry in *RXDID*.

7.6.3.3 ExtStat Field (FlexiFlags.1 Status Overlay)

Table 7-6 details the contents of the *ExtStat* field.

Table 7-6. ExtStat Field (4b) (When Enabled by ProfileID in FlexiFlags.1)

Bit	Field Mnemonic	Width (Bits)	Description
0	Ext_UDP_0	1	**Outer UDP Checksum Equals 0** — Identical to the field definition in the legacy descriptor format.
1	Int_UDP_0	1	**Inner UDP Checksum Equals 0** — Identical to the field definition in the legacy descriptor format.
2	RECIPE	1	**RDPU Recipe Error** — Identical to the field definition in the legacy descriptor format.
3	OVERSIZE	1	**Oversize Packet Error** — Identical to the field definition in the legacy descriptor format.

7.6.4 RXDID Descriptor Builder Profiles

For each packet, the format of the receive descriptor and particularly the contents that go into fields of the descriptor, is determined by the RXDID Descriptor Builder profile selected for the packet.

The E810 provides 63 x Descriptor Builder Profiles for programming different descriptor formats, including the 16-byte and 32-byte legacy descriptor formats that are preassigned:

- **Descriptor Builder Profiles #0** — Generates the legacy 16-byte descriptor format, identified by *RXDID*=0 in the descriptor structure delivered to software.
- **Descriptor Builder Profiles #1** — Generates the legacy 32-byte descriptor format, identified by *RXDID*=1 in the descriptor structure delivered to software.

Note: *RXDID*=7 is reserved and should not be used (in legacy descriptors, *MIRR*=7 means "mirror packet", and that's the only way to distinguish it from flex descriptors).

The Descriptor Builder Profile selected for each packet is derived either from a pipeline action that sets the *RXDID* MDID and its priority, or from the Flex Queue Context that defines an *RXDID* and priority per each queue. Conflicts are resolved through the standard MDID priority resolution scheme described in Section 7.3.

The *RXDID* field in the Flex Descriptor structure identifies the profile. Hence, the descriptor format delivered to software, in particular the contents of fields and flags.

The definition of a new Flex Descriptor format is as follows:

- Allocation of a new Descriptor Builder Profile entry.
- Programming the Descriptor Builder Profile entry:
  - FlexiMD contents (up to 6 x FlexiMD in 32-byte Flex Descriptors). Each MDID can provide one of the following:
    - MDID metadata from the pipeline (for example, opaque metadata from classification results).
    - Extracted fields from the packet (either header or payload up to the parsing depth of 504 bytes. See Section 7.6.7).
    - Pointers to locations of interest in the packet (for example, offset to the outer IPv4 header. Section 7.6.8).
  - FlexiFlags contents (up to 18 x FlexiFlags in 32-byte Flex Descriptors).

7.6.4.1 Programming RXDID Receive Descriptor Profiles

The Receive Descriptor Builder provides 64 x *RXDID* profile entries. The first two entries are preassigned for the legacy descriptor formats.

The configuration of a profile entry is done as follows:

For FlexiMD (n=0..5):

- GLFLXP_RXDID_MD[n].*ProtID_MDID*:
  - This field holds either a ProtocolID value or an MDID value. It's decoding is based on *RXDID_opcode* field.
    - ProtID (ProtocolID): For reporting its offset or for extracting packet bytes from this offset.
    - MDID: For reporting the MDID metadata contents.

- GLFLXP_RXDID_MD[n].*extraction_offset*:
  - Relevant only if *ProtID_MDID* field is a protID.
  - Used as offset within ProtID to extract bytes from the packet.
  - Its usage is based on *RXDID_opcode* field.
- GLFLXP_RXDID_MD[n].*RXDID_opcode*:
  - The opcode selects the type of report that goes into FlexiMD[n] in the Flex Descriptor:
    - 00b = "Fixed Value": Report the value from protID_MDID field itself (for debug purposes).
    - 01b = "Metadata Offset": Report the selected MD word from the internal FlexiPipe buses.
    - 10b = "Extraction Offset": Extract 16 bits (2 bytes) from offset location (ProtocolID offset + byte offset) in the packet.
    - 11b = "Protocol Offset": Report the offset to the selected ProtocolID.

For FlexiFlags.0 (6 bits) / FlexiFlags.1 (4 bits) / FlexiFlags.2 (8 bits) (n=0..17, total 18 bits):

- General:
  - The Receive Flex Descriptor includes 3 x FlexiFlags fields providing a total of 18 flags:
    - flexiflags0[5:0]
    - flexiflags1[3:0]
    - flexiflags2[7:0]
  - Note that the fields *flexiflag_4n*+2/*flexiflag_4n*+3 are invalid in last index as there are only 18 flags, not 20.
- GLFLXP_RXDID_FLAG [n].*flexiflag_4n/4n+1/4n+2/4n+3*:
  - Determines the flag index to be reported in flexiflag[4n/4n+1/4n+2/4n+3] from the FLG64 64 x flags available in the pipeline.

FlexiFlags.1 provides additional indications from RDPU that can override the regular flag selections:

- GLFLXP_RXDID_FLAGS1_OVERRIDE.flags1_override:
  - If Bit 0 is set, EXT_UDP_0 indication from RDPU replaces the value of flexiflags1[0].
  - If Bit 1 is set, INT_UDP_0 indication from RDPU replaces the value of flexiflags1[1].
  - If Bit 2 is set, RECIPE_ERROR indication from RDPU replaces the value of flexiflags1[2].
  - If Bit 3 is set, OVERSIZE indication from RDPU replaces the value of flexiflags1[3].

  Note: Further description of these indications from RDPU can be found in the legacy descriptors chapter section.

A special case is the support for legacy descriptors. For those, the configuration convention is:

- *RXDID* entry#0 is used for legacy 16-byte descriptors, and RXDID entry#1 is used for legacy 32-byte descriptors.
- Accordingly, the setting for the CSRs for entries 0 and 1 are:
  - GLFLXP_RXDID_MD.*protID_MDID*[0]=56 (HASH Low)
  - GLFLXP_RXDID_MD.*protID_MDID*[1]=57 (HASH High)
  - GLFLXP_RXDID_MD.*protID_MDID*[4]=5 (FDID Low)

— GLFLXP_RXDID_MD.*protID_MDID*[5]=6 (FDID High)

— GLFLXP_RXDID_MD[0].*RXDID_opcode*=01b (signaling MDID)

— GLFLXP_RXDID_MD[1].*RXDID_opcode*=01b (signaling MDID)

— GLFLXP_RXDID_MD[4].*RXDID_opcode*=01b (signaling MDID)

— GLFLXP_RXDID_MD[5].*RXDID_opcode*=01b (signaling MDID)

— All other fields are don't care for entries 0 and 1.

7.6.5 Receive Flex Queue Context

The E810 Flex Descriptor enhances the queue context with additional fields on top of the legacy queue context, as illustrated in Figure 7-9.

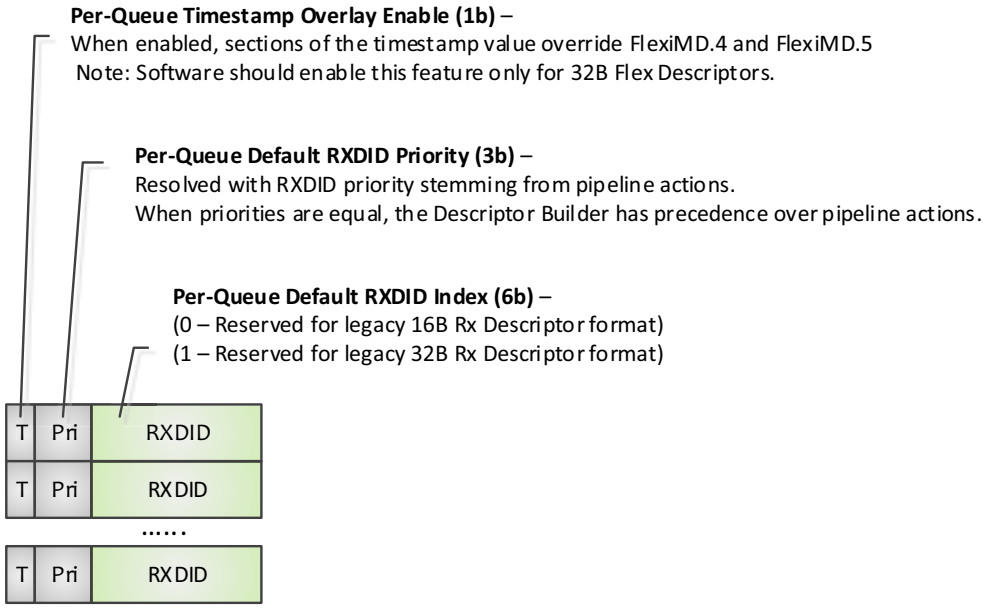


Figure 7-9. Receive Flex Queue Context

7.6.5.1 Programming the Receive Queue Context for Flex Descriptors

Programming of the additional Flex Queue Context is done via the QRX_FLEX_CNTXT CSRs, as follows:

- QRXFLXP_CNTXT.*RXDID_idx* — Determines the default *RXDID_idx* for this Q.
- QRXFLXP_CNTXT.*RXDID_prio* — Determines the default priority for the default *RXDID_idx* for this Q. This value is compared against RXDID priority from the RCU pipe. If this one is equal or higher, it wins, and the *RXDID_idx* for this pkt is determined by QRX_FLXP_CNTXT.*RXDID_idx*. Otherwise, it is determined by the value from the RCU pipe.
- QRXFLXP_CNTXT.*TS* — Determines whether timestamp reporting per Rx packet is enabled for this Q. If enabled, the *TS.H1*/*TS.H0*/*TS.L* fields in the Rx-Descriptor contains the timestamp for this packet. Bit 0 of the *TS.L* field is the "valid" bit for this timestamp.

7.6.6 Timestamp Overlay

The E810 provides a Timestamp Overlay feature for conveying timestamp information from receive ports to the host. The feature is enabled per destination Q via the Flex Queue Context structure, as described in Section 7.6.5.

The 40-bit timestamp value generated by the receive port is populated into fields in the 32-byte Receive Flex Descriptor as illustrated in Figure 7-10. Description of the Receive Flex Descriptor fields in provided in Section 7.6.3.

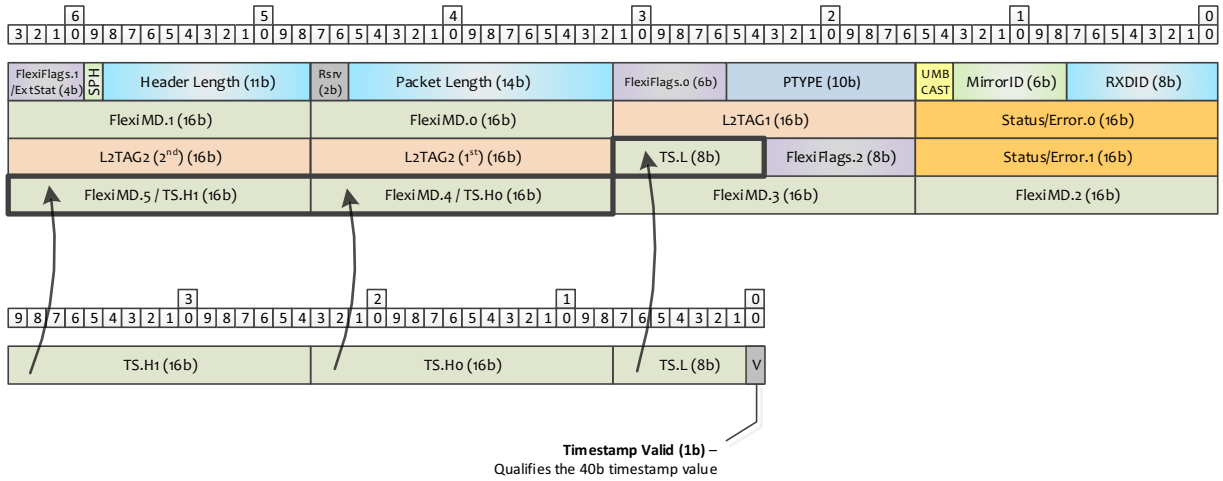


Figure 7-10. Timestamp Overlay in 32-Byte Receive Flex Descriptor

Note: The Timestamp Overlay feature is available only in 32-byte Flex Descriptors. The user should program the RXDID Profile for that queue accordingly, so that 32-byte Flex Descriptors are generated for packets arriving at that queue. The user should also enable the Timestamp Overlay feature in the Flex Queue Context as described in Section 7.6.5.

Note: The LSb (Bit 0) of the *TS.L* field is the timestamp valid bit, qualifying the timestamp value.

When the QRX_FLXP_CNTXT.*TS* bit in the Flex Queue Context is set, the E810 overlays the 40-bit timestamp value into these fields of the 32-byte Flex Descriptor:

- • TS.H1 (16 bits) — Replacing FlexiMD.5
- • TS.H0 (16 bits) — Replacing FlexiMD.4
- • TS.L (8 bits) — LSb (Bit 0) is the timestamp valid bit

The corresponding FlexiMD.5 and FlexiMD.4 metadata selected by the RXDID Profile for that packet are overwritten and replaced with sections of the timestamp value.

Note: When TimeStamp Overlay is enabled for particular queue in the Flex Queue Context, all 32-byte Flex Descriptors associated with packets arriving at that queue contain the timestamp overlay fields, regardless of whether the timestamp value is valid. When a queue is programmed to receive timestamps, software should evaluate the LSb (Bit 0) of *TS.L* of the 32-byte Flex Descriptor to infer whether a timestamp is valid for that packet.

7.6.7 Field Extraction into the Flex Descriptor

The E810 provides programmable field extractions from the packet into the Receive Flex Descriptor. Fields can be extracted either from the packet header or from the payload, up to the maximum parsing depth (504 bytes). Packet fields are extracted into the FlexiMD containers of the Receive Flex Descriptor. The number of field extractions provided depends on the size of the Receive Flex Descriptor:

- 16-byte Receive Flex Descriptors provide up to 2 x FlexiMD fields that can be used for field extractions.
- 32-byte Receive Flex Descriptors provide up to 6 x FlexiMD fields that can be used for field extractions.

Extraction width is always 16 bits. Software can opt to use smaller parts of the extraction (for example, when a smaller 8-bit is required.

Note: Any extraction of fields to the receive descriptor is done before any modification potentially done on the packet (for example, VLAN stripping) with the exception of CRC. Requesting the extraction of the CRC field to the receive descriptor is not allowed.

Each field extraction is defined by the following parameters in the RXDID Profile structure as described in Section 7.6.4:

- ProtocolID — Offset position generated by the parser.
- Offset — Relative byte offset from the ProtocolID position.

For example, to extract the destination port of a UDP packet, the user should program the extraction to use the ProtocolID of UDP with Offset=2 (bytes), as illustrated in Figure 7-11.

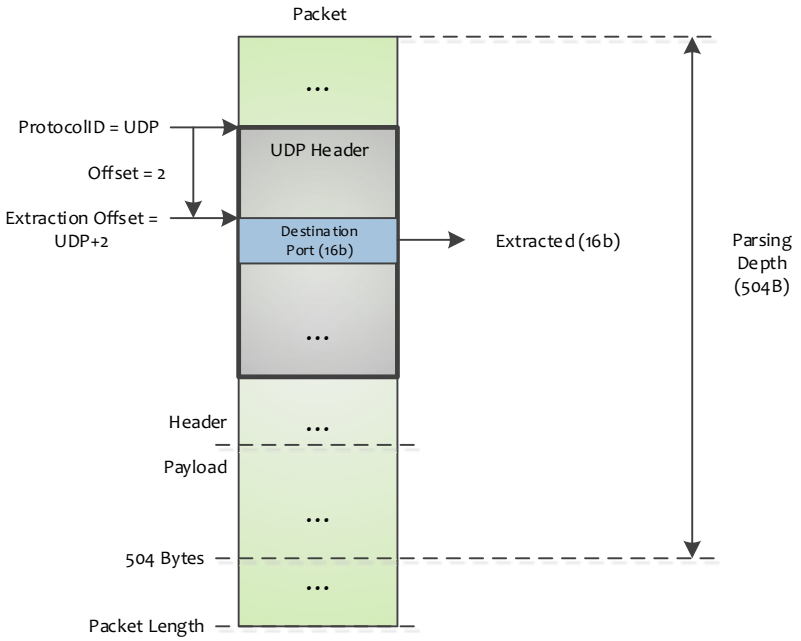


Figure 7-11. Field Extraction into the Flex Descriptor - Example

Note: Figure 7-11 illustrates a particular example in which the packet length is larger than 504 bytes. The packet header in this example is shorter than 504, allowing extraction from the payload.

The E810 indicates successful extraction per FlexiMD container by setting the corresponding XTR_MD.*#_VLD* bit in the Status/Error fields as described in Section 7.6.3.1 and Section 7.6.3.2. When the particular FlexiMD is programmed to contain an extraction result, but the XTR_MD.*#_VLD* bit is clear, software should disregard the value in the FlexiMD field.

The Receive Descriptor Builder validates for each extraction that the requested extraction offset is within packet boundaries. In case the extraction offset exceeds the packet boundary (that is, since the extraction works in word resolution, the byte offset is >{packet size - 2}), the Receive Descriptor Builder:

- Places the value 0xFFFF into the corresponding FlexiMD container.
- Clears the corresponding XTR_MD.*#_VLD* bit in the Status/Error field.
- Sets the Malicious Event exception flag.

Note: When using 16-byte descriptors, setting an out-of-bound extraction offset in FlexMD[5:2] (which is only relevant for 32-byte descriptors) also sets the malicious exception flag. Therefore, even when using 16-byte descriptors, the extraction offsets programmed for FlexMD[5:2] should contain valid offsets (though not used).

7.6.8 Pointers to Location of Interest

The FlexiMD metadata containers in the Flex Descriptor can be programmed to convey pointers to locations of interest in the packet, such as offsets to packet headers. Programming of the feature is done via the GLFLXP_RXDID_MD[n].*RXDID_opcode* CSR as described in Section 7.6.4.1.

For example, to provide the host with the IPv4 offset and the TCP offset of the packet, the RXDID profile entry is programmed to assign 2 x FlexiMD containers, as illustrated in Figure 7-12.

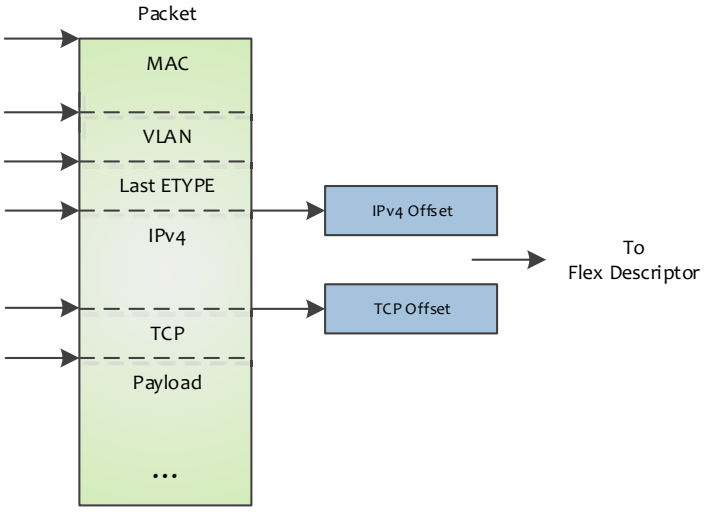


Figure 7-12. Field Extraction into the Flex Descriptor - Example

Note: Any offset report generated by the Parser can be placed into the Flex Descriptor. Offset reports normally point to protocol headers, however the Parser can be programmed to report offsets of any other location of interest such as TLV headers, option fields, and so on.

7.7 Programmable Parser

7.7.1 Introduction

The E810 provides a software programmable Parser (Analyzer) capable of supporting a wide range of well-known and proprietary protocols. The Parser examines the traffic that enters the pipeline, retrieves search parameters from the packet and from associated packet context, and then generates additional context based on certain packet attributes. This context is further used by other packet processing modules in the pipeline to associate the packet with a flow, software hints, and so on.

The factory parsing program loaded into the Parser from NVM supports a large set of frame formats that are common in various networking applications. These formats are provided by the E810 out of the box, not requiring any additional installation or programming.

A detailed description of the Factory Parsing Program appears in Appendix A.

These native formats include various protocols such as:

- L2: VLAN
- L2.5: MPLS
- L3: IPv4, IPv6
- L4: TCP, UDP, ICMP
- Overlay Network Formats: VXLAN, NGE, GRE
- Service chaining: NSH

7.7.1.1 Parsing Policy - Parse Graph

A parse graph is a representation of permitted protocol sequences in the network. The parse graph is composed of nodes and arcs. Each node represents a state, normally a protocol header. It is associated with a {NodeID} identifier. An arc represents traversal between nodes, normally between protocol headers. It originates from a source {NodeID} to a target {NodeID}.

Figure 7-13 illustrates a simple parse graph.

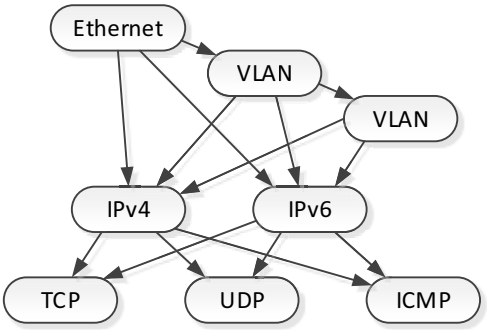


Figure 7-13. Simple Parse Graph

The simple parse is made from eight nodes and 14 arcs. The nodes in this parse graph correspond to protocols (Ethernet MAC header, VLANs, IPv4, IPv6, TCP, UDP, ICMP). Traversal conditions between nodes can be unconditional (that is, always take, or conditional (for example based on a protocol identifier in the protocol header such as the protocol field in an IPv4 header).

An example of an advanced parse graph scenario is illustrated in Figure 7-14.

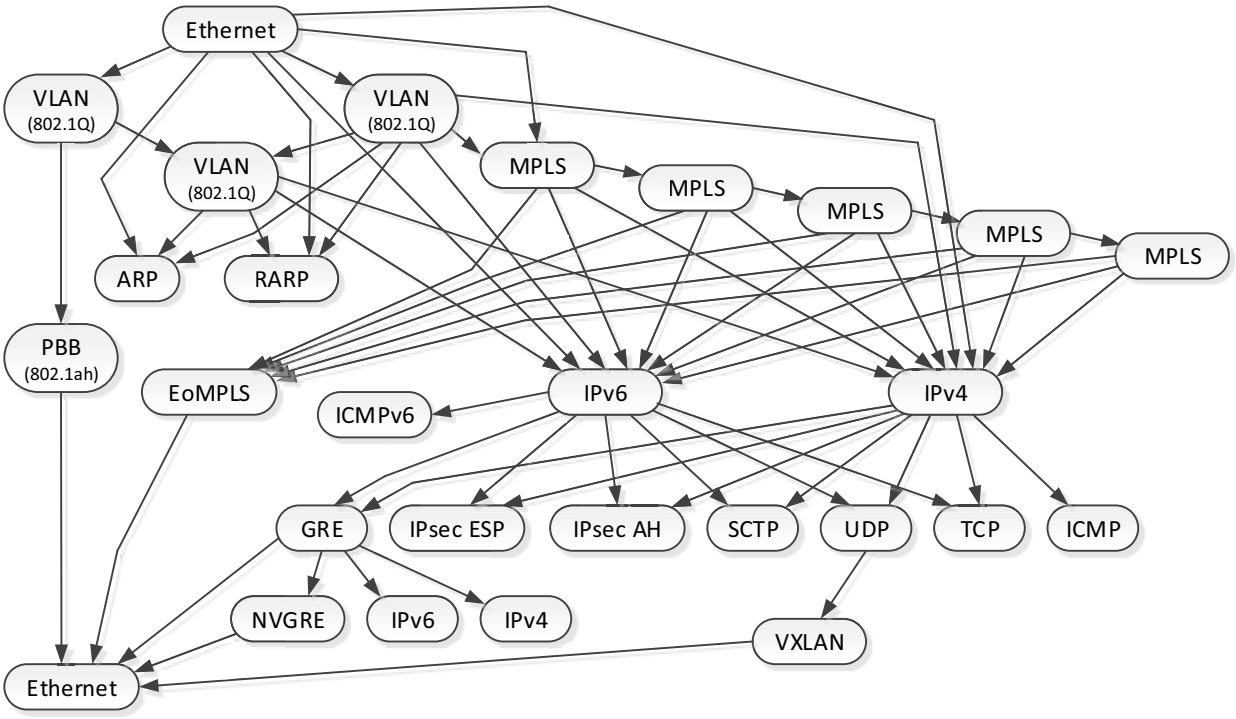


Figure 7-14. Advanced Parse Graph

7.7.1.2 Parsing Depth

The E810 Parser processes packet headers included in the first 504 bytes of the packet. The extensive depth covers complex encapsulation scenarios found in Cloud, Overlay-Networks, and Comms applications.

Note: Though the parsing depth is 504 bytes, the parser has an additional "protocol parsing depth" limitation allowing it to process up to 16 protocols (as specified in the parse graph) including the payload as the last protocol ID (the "payload" protocol ID is always be reported as the last protocol).

7.7.1.3 Parsing Profiles

The FlexiParser provides 16 x Parsing Profiles that determine the parsing execution flow.

A Parsing Profile determines:

- Initial PC (Program Counter) — Where the parsing program begins execution.
- Initial HO (Header Offset) — The offset location in the packet from where parsing starts.
- Initial NodeID — The root node of the parse graph selected.
- Register load — CPU General Purpose Registers (GPRs) loaded with pipeline metadata.

The structure of the profile table is shown in Table 7-7.

Table 7-7. Parser Profile Table

Field	Width	Description
TSR	8	Initial value for the TSR register.
HO	9	Initial value for the HO register.
NPC	8	Initial value for the NPC register.
NID	11	Initial value for the NID register.
Control domain	3	Control domain identifier. Used for marker PTYPE TCAM look up operation.
GPR_A_ctrl	1	General purpose register A Control: 0b = Copies constant value to the GPR. 1b = Copies a certain field from interface to GPR_A_ID: GPR_A_ID = MDID_A[MDID_A_start + MDID_A_len -1 : MDID_A_start] MDID_A_target_GPR = GPR_A_DATA
GPR_A_DATA	16	General purpose register A Data: Bits 0:4 = MDID_A Bits 5:8 = MDID_A_start Bits 9:12 = MDID_A_len Bits 13:16 = Reserved
GPR_A_ID	4	General purpose register A ID.
GPR_B_ctrl	1	General purpose register B Control.
GPR_B_DATA	16	General purpose register B Data.
GPR_B_ID	4	General purpose register B ID.
GPR_C_ctrl	1	General purpose register C Control.
GPR_C_DATA	16	General purpose register C Data.
GPR_C_ID	4	General purpose register C ID.
GPR_D_ctrl	1	General purpose register D Control.
GPR_D_DATA	16	General purpose register D Data.
GPR_D_ID	4	General purpose register D ID.
Flags initial value	64	Initial values for flags register.

7.7.1.4 PTYPE Generation

The FlexiParser generates a 10-bit PTYPE (Packet Type) that can be derived from:

- **Terminating NodeID** — When parsing stops at a NodeID that has an attribute of a terminating NodeID, the PTYPE is taken from the terminating NodeID PTYPE table.
- **Marker Sets** — When parsing stops at a NodeID that does not have an attribute of a terminating NodeID, the PTYPE is derived from the Marker Set Rules Table.

7.7.1.5 Programming Flow

The following sections describe the programming flows of the Parser.

7.7.1.5.1 Initialization

Following a device core reset the factory parsing program is loaded from NVM. A detailed description of the Factory Parsing Program appears in Appendix A. The parsing program includes complete configuration of the Parser, ready to run, including:

- Program code (iMEM)
- Protocol templates (TCAM)
- ProtocolIDs
- Parse Graph (PG)
- PTYPEs
- Well-Known/Public CDs (CD0+) and Profiles
- Misc tables/regs (e.g., ALU translation tables)

The parsing program carries a version number that the driver can query via a firmware AQC.

The factory parsing program loaded from NVM is initially unlocked and managed by firmware as a shared resource ownership using existing firmware infrastructure. A PF driver can obtain a lock on pipeline programming to update it via an ACQ.

The sequence is as follows:

1. Parsing image loaded from NVM.
2. First PF driver obtains lock on pipeline programming via ACQ.
3. PF driver queries version of parsing program via ACQ.
4. PF driver can choose to update the parsing program.
5. Firmware keeps timeout to release lock if PF driver fails/crashes.
6. When update is completed, the parsing program is available with updated version number.
7. PF driver continues programming the rest of the pipeline until done.
8. PF driver releases lock (the firmware provides a timeout fall-back procedure).
9. Firmware prevents further updates until next core reset or after the last PF driver goes down.
10. All PF drivers use the same parsing program.

Note: Manageability traffic is disabled during the update of the parsing program.

7.7.1.5.2 Configuration Access to Parser Resources

The Parser exposes indirect interface for reading and writing tables or execute table lookups using the following set of registers:

- GLGEN_ANA_CFG_CTRL
- GLGEN_ANA_CFG_RDDATA[0..15]
- GLGEN_ANA_CFG_WRDATA

7.7.1.5.2.1 Write Sequence

1. Before writing a line, firmware writes the GLGEN_ANA_CFG_CTRL register to point to the target line in the target table.
2. Firmware writes the GLGEN_ANA_CFG_WRDATA register multiple times until a complete line is accumulated.
3. The Parser accumulates the 32-bit CSR writes into a wide data line. When it detects that a full line is accumulated, it writes it to target line in memory.
4. In case firmware continues writing the WR_DATA (without writing the CTRL register first), the Parser continues to write the next line in the same memory.

7.7.1.5.2.2 Read Sequence

1. Firmware writes the GLGEN_ANA_CFG_CTRL register to point to the target line in the target table.
2. Firmware reads the relevant data directly from GLGEN_ANA_CFG_RDDATA[0..16] registers.
3. Before reading the next line, firmware must set the GLGEN_ANA_CFG_CTRL register again.

7.7.1.5.3 In-Service Programmability

The Parser provides in-service programming of the following resources:

- Add/Remove of PG arc
- Add/Remove of TCAM entry
- Add/Remove/Change of Profile

Note: It is assumed that the associated PG nodes and iMEM instructions are already configured in the Parser database.

7.7.1.5.3.1 Add/Remove of PG Arc

The flow of adding/removing of PG is as follows:

1. Firmware writes the PGkey to GLGEN_ANA_CFG_LU_KEY.
2. Firmware initiates PG lookup via GLGEN_ANA_CFG_CTRL.
3. The Parser hardware returns:
   - The CAM hit result, the CAM bank number (when not in spill buffer), and the CAM line index in GLGEN_ANA_CFG_SPLBUF_LU_RESULT and GLGEN_ANA_CFG_HTBL_LU_RESULT.
   - TCAM unit 0 capacity - GLGEN_ANA_CFG_RDDATA 0.

- TCAM unit 1 capacity - GLGEN_ANA_CFG_RDDATA 1.
- TCAM unit 0 bin vector - GLGEN_ANA_CFG_RDDATA 2.
- TCAM unit 1 bin vector - GLGEN_ANA_CFG_RDDATA 3.
- TCAM unit 0 HASH value - GLGEN_ANA_CFG_RDDATA 4.
- TCAM unit 1 HASH value - GLGEN_ANA_CFG_RDDATA 5.
- Spill buffer capacity vector - GLGEN_ANA_CFG_RDDATA 7/8.

4. Firmware initiates or resets the PG action memory using memory write sequence.
5. Firmware initiates or resets the PG key using memory write sequence.

7.7.1.5.3.2 Add/Remove of TCAM Entry

The flow of adding/removing of TCAM entry is as follows:

1. Firmware writes an action to a well-known address in the TCAM action memory using the memory write sequence.
2. Firmware writes a key to a well-known address in the TCAM key using the memory write sequence.

7.7.1.5.3.3 Entry Change in Profile Table

Table content change happens only for a Profile that its packets are drained from pipe.

Before changing content:

1. Firmware routes the modified profile N to a stable default profile by writing the default Profile ID to index N in the P2P table.
2. Firmware waits 10 μs. During this time period it is guaranteed that:
   - Packets processed in pipe before the change are drained towards the reorder logic.
   - New packets enter the pipe with the default profile.
3. Firmware does table changes for the reset profile using memory write sequence.
4. Firmware enables the modified profile by writing “N” to the index N in the P2P table.

7.8 Switch (Binary Classifier)

7.8.1 Features

Table 7-8. Binary Classifier Features

Description
Logical Switching Tables • Supports up to 64 different switching lookups/recipes (all actions included). • Supports up to 10 concurrent switching lookups per specific packet type @ 90 Mpps. • Each lookup can contain up to five 16-bit tuples. Lookups can be chained to create longer recipes. • Support up to 32K lookup entries. The actual number depends on the length of the recipe and the utilization of the hash.
Built in recipes loaded from NVM to implement basic NIC functionality • Supports simple L2 switching including VEB and VEPA modes. • Supports default VSI per VEB (using default, direction recipe). • Supports flooding action per VSI. • Anti spoofing.
Sample use cases to be supported • Multi Channel VEPA (STag based). • Tenant Forwarding (NVGRE, Geneve, and so on). • Q-in-Q forwarding. • Private VLAN. • Local VLAN. • Floating VEB.
Support for multiple switch instances • Allow classification of traffic to different switch instances based on source or tag in packet. • Tag can be up to four words long from anywhere in header. • Support up to 64 tag based switch IDs.
Allows Tx-Descriptor from trusted driver to define the destination of a packet. Allows the following options: • Send To LAN only. • Do not send to LAN. Let switch define the destination within loopback. • Send to a specific VSI
Actions - support multiple type of actions per recipe • Forwarding - to VSI, VSI list, queue group within a VSI, or queue within a VSI. • Pruning. • Statistics. • Support up to 8K basic actions that can be chained in up to four actions. • Multiple forwarding actions can be prioritized or merged.
Support for different switching Profiles • Applicable recipes can be selected according to source of packet, packet type, and other packet parameters. • Support bypass of classifier for selected packet profiles. • Allow trusted software to fix destination of packet.
Egress Rules - support additional actions on egress copy of a packet • Egress mirroring. • Source pruning. • Limit decrypted traffic to function owning the SA.
Support up to 768 VSIs. • Allow replication to multiple VSIs.

Table 7-8. Binary Classifier Features [continued]

Description
Statistics • Per VSI statistics. • Up to 32 per VEB statistics block. • Up to 128 per VLAN/UP statistics block.
Mirroring • Ingress mirroring (according to packet source). • Egress Mirroring (according to packet destination). • VLAN/VEB mirroring. • Event based mirroring. • Mirroring destination to local VSI only.
Storm Control • Per port broadcast and multicast storm control.
Programming API • Allow Software to add or remove switching recipes on the fly. • Allow Software to classify traffic to profiles on the fly.
Manageability forwarding. • Supports forwarding of packets to the BMC or the internal EMP based on the MDEF filters described in Section 12.4.

7.8.2 Binary Classifier (Switch) Block Diagram

Figure 7-15 describes the switch blocks:

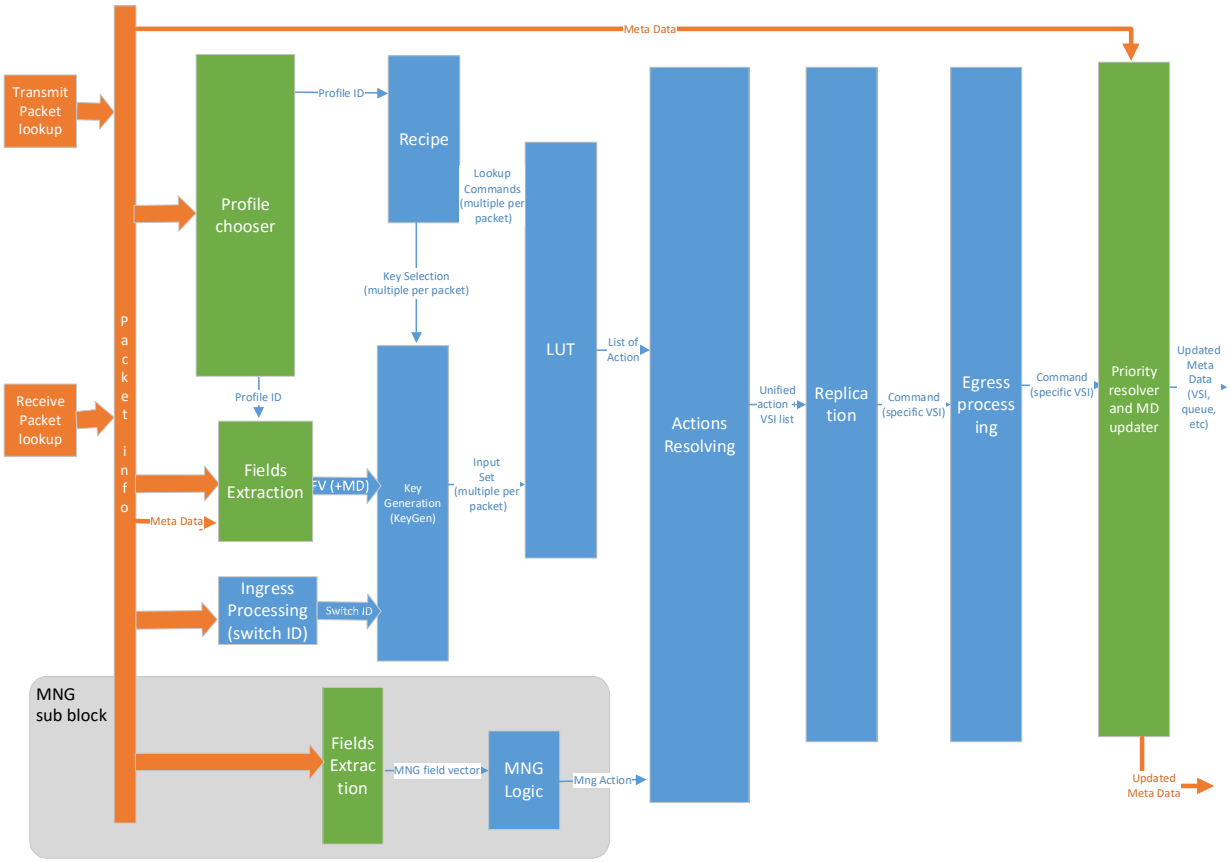


Figure 7-15. Switch Block Diagram

A packet entering the E810 switch goes through the following steps, depicted Figure 7-15:

1. **Profile Selection** — A Profile ID is selected based on metadata of the packet (typically, the packet type and the source of the packet). The Profile ID later defines the set of recipes and the input set that is relevant for each packet.
2. **Switch ID Generation** — The header and metadata of the packet are used to define a switch ID for the packet. The switch ID associates a packet with a specific switch instance. The switch ID is used in subsequent steps to identify lookup entries that are relevant to this packet.
3. **Field Extraction** — Based on the Profile ID, a set of fields from the header are extracted and organized as a "field vector". The extracted fields should be all the fields relevant for the different recipes lookups.
4. **Lookups**:
    a. Recipes selection — Based on the Profile ID, a set of lookups to apply to the packet's field vector are selected.

b. For each recipe, the Key Generator (KeyGen) selects from the packet's field vector the parts of the header needed to create this lookup and presents the results to the Lookup Table (LUT).

c. The LUT is a binary CAM containing the various lookup values and actions. The output of this stage is a set of actions to apply to the packet.

5. **MNG Sub-Block (Manageability Lookup)** — In parallel to the host forwarding lookup, filters specific to traffic that need to be forwarded to the BMC via sideband are applied to implement NC-SI and other pass-through forwarding rules. This stage can add an additional action to the set previously defined. This block has its own profile selection unit.
6. **Action Resolving** — After all the actions are created, unified actions are created. The output is a list of VSIs that should receive the packet and other actions such as statistic actions and others.
7. **Replication** — For each VSI in the resulting VSI list, a command to replicate the packet is created.
8. **Egress Processing** — For each copy of the packet, egress rules are applied. Applied actions can be egress mirroring.
9. **Priority Resolver and Metadata Updater** — The resulting commands are forwarded to the next stages of the pipe. As part of this the decision of the switching blocks are merged into the global metadata bus.

7.8.3 Control Domain and Profile Selection

The profile selection block is described in Section 7.2.2. The usage of this block in the switch context is to select a profile based on the packet type and based on the source of the packet. Selection based on the packet type is used to reduce the lookups to the ones relevant to this packet type. Selection based on the packet source is used to assign different recipes for different networks.

There are up to 256 profiles in the binary classifier, and 512 rules in the TCAM to select them.

A profile is be selected according to the following parameters. Some of them are used as part of the current implementation and some are reserved for future expansions:

- Packet Source:
  - — Transmit - VSI Number
  - — Receive - Physical port# or logical port ID
- Packet Source Type:
  - — Receive - from LAN
  - — Loopback - from Host
  - — Loopback - from EMP/BMC
  - — Transmit - query

  Note: The packet source type and packet source are used to create the first input of the control domain selection, thus allowing selection of a control domain according to the VSI that sent the packet or according to the port on which the packet was received.
- Packet Type — This field allows different profiles for different packet type groups, thus allowing reduction of the number of recipes that can be applied to a single packet, and improving the performance of the switch.

- Flags:
  - Bypass switch command, bad packets, SWPE — Fields that allow a bypass of the switch for packets that do not require a switch decision via the lookup tables.
  - Software command — This field is be used to prevent regular traffic from matching a profile, while allowing software commands to still match the profile (for example, to delete recipes).
- Other optional fields not currently used, but that can impact the profile:
  - TC
  - Other fields can be defined later by software.

7.8.4 Ingress Pre-Processing

7.8.4.1 Field Extractor

The Fields Extractor is a standard block that extracts a local Field Vector (FV) from the packet info. The list of fields that are extracted is based on the selected Profile ID.

The configuration of this block should be that all the fields used by any of the recipe selected by this Profile ID are represented in the generated field vector.

See Section 7.5 for programming of the extraction vector and encoding of the Protocols.

The resulting vector can be up to 48 words for the main switch and 32 words for the manageability decision. See Section 7.8.7 for details of the manageability filtering Fields Extractor.

The default vector is defined in Section 7.8.11.3.2.

7.8.4.2 Switch ID Creation

The ingress switch ID is an identifier of the virtual VEB to which the packet belongs, and it is used as an input to the LUT.

The switch ID is based on the source of the packet (VSI, port or logical port).

The input to this block is the metadata bus and the field vector. The relevant information taken from these buses is the packet source:

- For Transmit - VSI Number
- For Receive - Physical port# or logical port ID

The following data is defined in the block:

- A per-VSI switch ID — Used to identify the switch ID of a packet based on the VSI that sent the packet. This is part of the VSI context and is stored in the VSI_SWITCHID array (*SWTCHID*[7:0] field).
- A per logical/physical port switch ID — Used to identify the default switch ID of packets. Stored in the GL_VP_SWITCHID array (*SWITCHID*[7:0] field).
  - The selection between physical or logical port is done according to the GL_SWT_SWIDFVIDX.*PORT_TYPE* field (0b = Physical, 1b = Logical).

The switch ID is created according to the following logic:

```
if (packet is Tx (loopback or query)) { switch ID = VSI Switch ID}
else if (packet is rx) { switch ID = logical port/physical port Switch ID}
```

The output of the Block is switch ID[7:0] and it is added as part of the Field vector.

Per switch ID, a GLSWID_STAT_BLOCK register defines if one of 32 VEB statistic block is associated with this switch ID. The GLSWID_STAT_BLOCK.*VEBID* field defines the stat block ID to which the switch ID is associated. A switch ID cannot be associated with any switch block (GLSWID_STAT_BLOCK.*VEBID_VALID* = 0).

The switch ID extracted as described above should be part of the Field Vector described in Section 7.8.4.1 to enable differentiation between lookup entries of different VEBs.

7.8.5 Switching Engine

7.8.5.1 Switching Recipes and Key Generation (KeyGen)

The switch recipe block is a set of 64 lines that describes the lookups to do on the field vector. Each recipe can be based on multiple lookups and multiple recipes can be applied to a single packet. The set of recipes to apply on a given packet is defined according to the Profile ID using a per Profile ID bitmap defining which lookups to apply.

In addition, another table maps the dependencies between recipes. For each recipe, there is a bitmap of all the recipe entries that are part of the decision tree of this recipe.

Each lookup line can be defined as a root of a recipe or a line that creates an intermediate lookup used for another calculation. A root line can also serve as an intermediate lookup of another root line. Only root lines points to an action.

The intermediary results are stored back into the field vector and so can be used as part of the input set of subsequent lookups.

Lines that are used as intermediate steps of a recipe should be stored before the root of this recipe. See Figure 7-16 for the multiple lookup recipe diagram.

Per profile, up to 32 lookup lines can be selected, and 24 of them can be recipe roots.

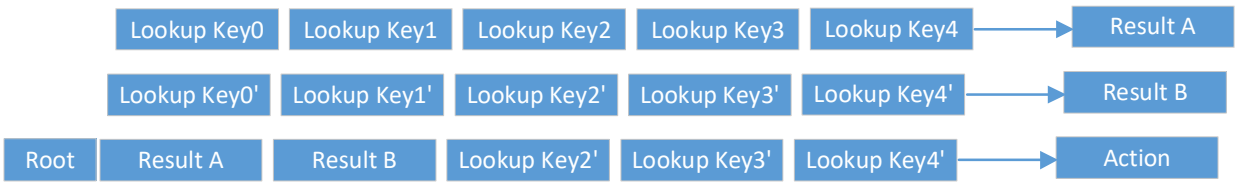


Figure 7-16. Multiple Lookup Recipe

Once a lookup line is chosen, the list of lookup words are used by the KeyGen block to build the part of the lookup key extracted from the packet header or the metadata. The generated key, the mask bits, and the RID are then looked up against the Lookup table.

The lookup lines are applied in the order they appear in the table. Hence, entries in the recipe table should be in order so that lines used as an intermediate lookup creates intermediary results before a line that uses the lookup result is run.

Each lookup line contains the following information:

Table 7-9. Recipe Content

Field	Byte Range	Width	LSB	MSB	Sub-Field Description
RID	Byte 0	6	0	5	Recipe ID (RID)[1].
		1	6	6	Reserved (for recipe expansion).
		1	7	7	isroot — Defines if this lookup line is a root of a recipe.
Lookup Index 0	Byte 1	6	8	13	First word to be compared to data from the Field Vector or the metadata. The value is an offset within the extracted Field vector. Unused fields should be masked in the *Mask* field.
		1	14	14	Reserved.
		1	15	15	Ignore Valid — If set, the field match even if the field is not valid. Should be set for VLAN entries where a value of zero with valid bit clear indicates an untagged packet that should be treated as VLAN = 0.
Lookup Index 1	Byte 2	6	16	21	Second word index in Field Vector.
		1	22	22	Reserved.
		1	23	23	Ignore Valid — If set, the field match even if the field is not valid.
Lookup Index 2	Byte 3	6	24	29	Third word index in Field Vector.
		1	30	30	Reserved.
		1	31	31	Ignore Valid — If set, the field match even if the field is not valid.
Lookup Index 3	Byte 4	6	32	37	Fourth word index in Field Vector.
		1	38	38	Reserved.
		1	39	39	Ignore Valid — If set, the field match even if the field is not valid.
Lookup Index 4	Byte 5	6	40	45	Fifth word index in Field Vector.
		1	46	46	Reserved.
		1	47	47	Ignore Valid — If set, the field match even if the field is not valid.
Mask	Bytes 6-15	16	48	63	Bit-mask of word 0 of the LU key. 1 means the bit is compared.
		16	64	79	Bit-mask of word 1 of the LU key.
		16	80	95	Bit-mask of word 2 of the LU key.
		16	96	111	Bit-mask of word 3 of the LU key.
		16	112	127	Bit-mask of word 4 of the LU key.
Result Index	Byte 16	6	128	133	Word index in which LU result will be placed.
		1	134	134	Reserved.
		1	135	135	Result Enable — LU yields result that should be stored in Field Vector.
Reserved	Bytes 17-19	24	136	159	Reserved.

Table 7-9. Recipe Content [continued]

Field	Byte Range	Width	LSB	MSB	Sub-Field Description
Action Control	Bytes 20-23	8	160	167	A bitmap of the join priorities of forwarding actions associated to this recipe. Relevant only for roots. The join priority defines which priorities are joined by this recipe. See Section 7.8.6.1 for details.
		4	168	171	The priority of forwarding actions associated to this recipe. Relevant only for roots.
		4	172	175	Reserved.
		1	176	176	Need_pass_l2 — If set, for the action of this recipe to be relevant, a recipe with *allow_pass_L2* bit set should match[2].
		1	177	177	Allow_pass_l2 — If set, allows recipes with *need_pass_L2* bit set to pass[3].
		1	178	178	Inverse Action — This bit is used to allow forwarding lookups to be used as anti-spoof (Tx) and source pruning (Rx) actions. See Section 7.8.5.1.1 for details.
		1	179	179	Reserved.
		2	180	181	Prune Index — Defines the enable bits to use it the VSI context to qualify a pruning action associated with this recipe. The VSI_SRCSWCTRL.*PRUNEENABLE*[3:0] and VSI_RXSWCTRL.*PRUNEENABLE*[3:0] are used to enable the pruning in Tx and Rx respectively.
		10	182	191	Reserved.
Default Action	Bytes 24-27	19	192	210	Default action according to the format described in Table 7-12. The default action can be a pointer to a large action or a single action.
		12	211	222	Reserved.
		1	223	223	Default Action valid.

1. For normal recipes, the RID should be equal to the index of the recipe. If *Inverse Action* flag is set, the RID might differ from the recipe index. In this case, lookup entries should not be populated with this recipe index. For non-root recipes, the RID must be zero.
2. This bit requires another recipe lookup with *Allow_pass_l2* set to match.
3. This bit identities this filter as an L2 filter, qualifying filters with *Need_pass_l2* set.

7.8.5.1.1 Inverse Action

The operation of anti-spoof of transmit packets is based on checking the SA or the {SA, VLAN} pair used to send the packet. If this SA or {SA, VLAN} pair where used as DA or {DA, VLAN} it would have to allow the VSI to receive the packet, then it can use them as SA or the SA, VLAN pair.

Similarly the operation of source pruning in receive can be based either on the source VSI as described in Section 7.8.6.5 or on the identification the SA or {SA, VLAN} pair as belonging to a VSI by identifying that if his SA or {SA, VLAN} pair where used as DA or {DA, VLAN} it would have allow the VSI to receive the packet.

This means that in the same MAC Address or MAC/VLAN pair could be used both to forward to a VSI or for anti-spoof or source pruning actions. To save the need for two entries for each of the MAC Address or MAC/VLAN pair in the lookup table, the inverse action mechanism is provided.

If inverse action bit in recipe (Bit 178) is set, any forwarding action to a single VSI or to a queue in the recipe is replaced with comparison of the source VSI of the packet with the VSI or the VSI matching the queue in the action for Tx traffic, and is replaced with a prune (AND NOT) for receive traffic. There is no support for inverse action when the forwarding action points to a list.

This is applied only if the action is a VSI forwarding action to a single VSI or a ToQueue action, either in a regular or large action.

If set, the action is enabled for Tx traffic using the VSI_SRCSWCTRL.*MACAS* bit and for Rx traffic using the VSI_RXSWCTRL.*MACVSIPRUNEENABLE* bit.

When setting this bit, the RID of the inverse action recipe should be equal to the RID of the matching forwarding recipe.

7.8.5.1.2 Sample Recipes

Table 7-10 provide a few examples of recipes (a letter in one of the words indicates usage of intermediary lookups, according to the result field in the intermediary line). Lines 0-10 are the basic recipes loaded from NVM.

Note: The layout of the recipes can be different if needed to optimize lookups. For example, if a part of a lookup is common to multiple lookups (for example, {switch ID, tenant ID, Key}), it might be worthwhile to have it a single lookup entry shared by multiple recipes.

Table 7-10. Sample Recipes

Recipe Index (RID)	Recipe	Priority (Joint Priority)[1]	Is Root	Word0	Word1	Word2	Word3	Word4	Result[2]
0	EtherType, Direction	4	Y	switch ID	EtherType	Direction	-	-	E
1	DA	2	Y	switch ID	MAC0	MAC1	MAC2	-	B
2	DA, VLAN	2	Y	switch ID	MAC0	MAC1	MAC2	VLAN	-
3	Packet Type, Direction (promiscuous)	1 (1,2,3)[3]	Y	switch ID	Packet Type	Direction[4]	-	-	-
4	VLAN[5]	N/A[6]	Y	switch ID	VLAN		-	-	-
5	Default, Direction	0	Y	switch ID	Direction	-	-	-	-
1	SA (inverse action)	N/A[7]	Y[8]	switch ID	SMAC0	SMAC1	SMAC2	-	-
2	SA, VLAN (inverse action)	N/A	Y[9]	switch ID	SMAC0	SMAC1	SMAC2	VLAN	-
8	EtherType, MAC, Direction	4	Y	switch ID	MAC0	MAC1	MAC2	E	-
9	Promiscuous, VLAN, Packet Type, Direction	1 (1,2,3)[3]	Y	switch ID	Packet Type	VLAN	Direction	-	-
10	Logical Port	0	Y	switch ID	Logical Port				
11	Inner MAC, Tunnel Type	3	Y	switch ID	IMAC0	IMAC1	IMAC2	Tunnel Type	A
12	Inner MAC, Inner VLAN, Tunnel Type	3	Y	A	iVLAN	-	-	-	-
13	Inner MAC, Inner VLAN, Tenant ID, Tunnel Type	3	Y	A	iVLAN	Key0	Key1	-	-
14	Inner MAC, Tenant ID, Tunnel Type	3	Y	A	-	Key0	Key1	-	-
15	Outer MAC, Tenant ID, Inner MAC, Tunnel Type	3	Y	B	A	Key0	Key1	-	-
16		N/A	N	D-IP-0	D-IP-1	D-IP-2	D-IP-3	D-IP-4	C
17	Application Dest IPv6	3	Y	switch ID	D-IP-5	D-IP-6	D-IP-7	C	-
18		N/A	N	S-IP-0	S-IP-1	S-IP-2	S-IP-3	S-IP-4	G
20	Application Dest IPv4	3	Y	switch ID	D-IP-0	D-IP-1	-	-	D

Table 7-10. Sample Recipes [continued]

Recipe Index (RID)	Recipe	Priority (Joint Priority)[1]	Is Root	Word0	Word1	Word2	Word3	Word4	Result[2]
21	Application Source IPv6, Inner MAC	3	Y	S-IP-5	S-IP-6	S-IP-7	A	G	-
22	Application Source IPv4, Inner MAC	3	Y	S-IP-0	S-IP-1	A	-	-	-
24	Application Dest IPv4, Application Source IPv4	3	Y	S-IP-0	S-IP-1	D	-	-	-
25	TCP/UDP Port	3	Y	switch ID	DPort	-	-	-	-
26	STag	1	Y	switch ID	STag	-	-	-	-
27	iVLAN	3	Y	switch ID	iVLAN				

1. If a joint priority is not provided, it is equal to the regular priority.
2. The letters in the Result column indicate indexes in the field vectors that are updated by a match of this recipe, and are used as inputs in subsequent recipes.
3. Assuming promiscuous additive to other filters.
4. For rules where a direction is used, it is defined by the type of rule added (T_LOOKUP_TX (packet sent by the driver) or T_LOOKUP_RX (packet received by the driver)).
5. Should allocate a Prune Index to this rule (assume Prune Index = 0 for the examples and for the NVM default package).
6. Assumption is that VLAN filter is used for pruning.
7. No priority for inverse actions.
8. Use same RID as DA table. Used for anti-spoof/source pruning.
9. Use same RID as DA, VLAN table. Used for anti-spoof/source pruning.

7.8.5.2 Lookup Table

The lookup table is a Binary-CAM with 32K entries accessed using the Key generated in the KeyGen. The masked lookup key is searched in the table. If a match is found, the result is provided to the action resolving block that registers an action to do on the packet (if this is a root lookup) and to the KeyGen that can update the Field Vector (if intermediate or root lookup). If a match is not found, a default action per recipe can be used.

The format of each entry in the lookup table is as follows:

Table 7-11. Field Lookup Entry Format

Field		Width	LSB	MSB
Action		19	0	18
Ref. Count		16	19	34
Pointer		16	35	50
Filter Key	FV Word 0	16	51	66
	FV Word 1	16	67	82
	FV Word 2	16	83	98
	FV Word 3	16	99	114
	FV Word 4	16	115	130
	RID	6	131	136
	Reserved	9	137	145
	IsRoot	1	146	146

7.8.5.3 Actions

There are three types of actions supported:

- Single action per lookup (Section 7.8.5.3.1).
- List of actions per lookup divided to three sections of single, double, and quad lists (Section 7.8.5.3.2).
- Default Action, if the lookup does not result in a match (Section 7.8.5.3.3).

Figure 7-17 shows the different types of actions:

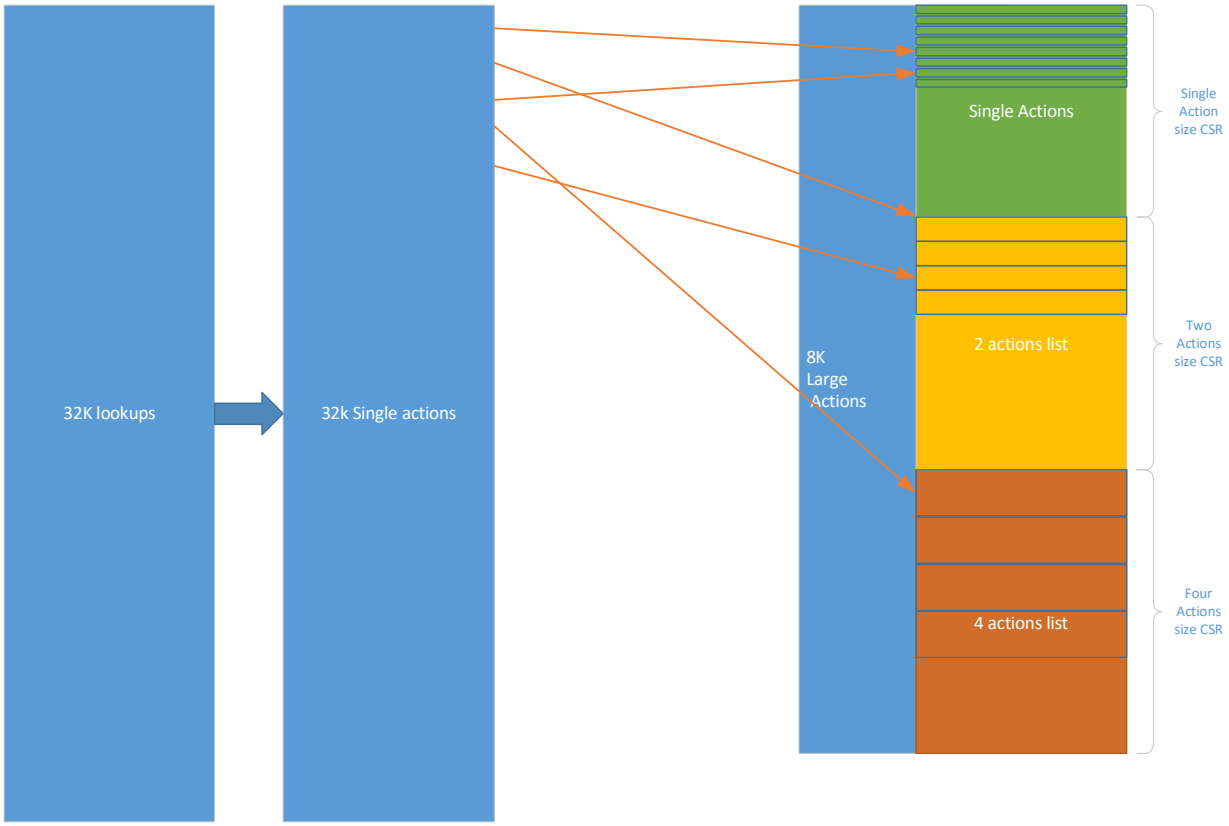


Figure 7-17. Single Actions and Large Actions

7.8.5.3.1 Single Action

For each of the 32K lookups, a single action can be applied if the lookup matches.

The following actions can be requested by a lookup entry:

- Forwarding actions (see Section 7.8.5.3.5)
- Pruning action (see Section 7.8.5.3.7)
  - — Prune the existing forwarding list to include only VSIs matching a VSI or VSI list (AND operation). This action can be applied to ingress traffic, egress traffic or both.

Note: The usage model is to allow a single VLAN entry to point to different lists for ingress and egress traffic.

— Prune the forwarding list not to include a VSI list (AND NOT operation).

- Statistic actions (see Section 7.8.5.3.9)

If a large action is needed, the single action can be replaced with a pointer to a list of actions described below.

Note: All actions are additive, apart from forwarding actions to a VSI or a queue, which are defined by priorities.

The format of the single action is described in Table 7-12.

Table 7-12. Single Actions Format

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
00b (VSI Forwarding)		LB EN[1]	LAN EN[2]	VSI/VSI_list										List	Reserved		Valid[3]	Drop[4]
01b (To Queue)		LB EN	LAN EN	Q Index[5]											Q Region Size			Q_PRI[6]
10b (Prune)		LB EN[7]	LAN EN	VSI_list										List = 1[8]	Egr[9]	Ing[10]	Prune T[11]	0b
10b (Pointer)		LB EN[7]	LAN EN[12]	Pointer to a large action. Points to the first action in the large action in the 8K range.													hasFWD[13]	1b
11b (Other Actions)		Reserved		Mirror VSI										Reserved			00b (Mirror)	
		Reserved															10b/11b (Reserved)	
		Reserved		Statistics Counter Index							Reserved						11b (Stat Count)	

1. Allows packet to be looped back.
2. Might prevent packet from being sent to the network.
3. The *Valid* bit should be set by the software device driver in all forwarding actions. Adding an entry with *Valid* = 0 is allowed if *allow_pass_L2* is set in the recipe, and only a hit in the table is needed with no real action.
4. If *Drop* is set, VSI/VSI_list is ignored.
5. The queue index is an absolute queue number in the 0-2K range and should be used as such in the lookup programming commands.
6. Defines queue priority.
7. *LB EN* should not be set for prune actions or for large actions not including a forwarding action.
8. Must be set. Direct pruning is only available according to a VSI list.
9. Apply pruning to egress traffic (traffic received by VSIs).
10. Apply pruning to ingress traffic (traffic sent by a VSI).
11. AND prune (0) or AND NOT prune (1).
12. *LAN EN* is relevant only if large action includes prune or forwarding actions.
13. Should be set if the large action includes a forwarding command.

7.8.5.3.2 Large Actions

To allow multiple actions to be activated from a single lookup, on top of the 32K per lookup actions, a table with 8K entries is provided. Each entry represents a single action, and up to four actions can be chained into an action list. If the original action points to one of these action lists, all the combined actions are applied.

These 8K are divided to three programmable sections:

- A section of single actions (to allow usage of new actions not available directly).
- A section of two actions lists.

- A section of four actions lists.

Each lookup can point to any section depending of the number of requested action. The size of each section is programmable from NVM or from software device driver at init time (before any extended action is programmed) via the following registers:

Table 7-13. Large Actions Table Allocation

Table Part	Register	Note
Single Action	GL_SWT_LAT_SINGLE.*BASE*[10:0]	Encoded as base/4
	GL_SWT_LAT_SINGLE.*SIZE*[10:0]	Encoded as size/4
Double Actions	GL_SWT_LAT_DOUBLE.*BASE*[10:0]	Encoded as base/4
	GL_SWT_LAT_DOUBLE.*SIZE*[10:0]	Encoded as size/4
Quad Actions	GL_SWT_LAT_QUAD.*BASE*[10:0]	Encoded as base/4
	GL_SWT_LAT_QUAD.*SIZE*[10:0]	Encoded as size/4

The allocation is common to all ports.

Note: As there is a single such table, usage of large actions might degrade performance, if applicable for a large part of the traffic and used multiple times on the same packet.

Any of the actions included in the list can implement one of the actions defined above. In addition, the following actions are available only as part of large actions:

- Generic Action to update metadata (see Section 7.8.5.3.8).
- Increase counter — Used to support event counters.

The format of each action in an action list is described in Table 7-14.

Table 7-14. Large Actions Entries

0:2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19:21	22:24
000b (VSI Forwarding)	VSI/VSI_list										List	Reserved		Valid[1]	Reserved			
001b (To Queue)	Q Index[2]											Q Region Size			Q_PRI[3]	Reserved		
010b (Pruning)	VSI_list										List = 1[4]	Egr[5]	Ing[6]	Prune[7]	Reserved			
011b (Mirroring)	Mirror VSI										Reserved							
101b (Generic Action)	Generic Value																Generic Offset	Generic Priority
110b	Spare																	
010b (Statistics)	Statistics Counter Index							Reserved										

1. The *Valid* bit should be set by the software device driver in any forwarding action.
2. Queue index is the absolute queue number (0-2047) and should be used as such in the lookup programming commands.
3. Defines queue priority.
4. Must be set. Direct pruning is only available according to a VSI list.
5. Apply pruning to egress traffic.
6. Apply pruning to ingress traffic.
7. AND prune or AND NOT prune.

7.8.5.3.3 Default Actions

Each entry in the recipe table associated with an RID (isroot = 1) can define a default action to take if a lookup was done but no match was found. For example a VLAN lookup default action might be to prune all VSIs that are doing VLAN filtering (not in ignore VLAN mode). For a MAC lookup, the default action might be null. The default action layout is described as part of the recipe in Table 7-9.

7.8.5.3.4 Valid = 0 Action

A recipe can use as sub-entries other root entries. In this case, the entry can match as part of the large lookup, but not induce a match of an actual filter. For example, a MAC/VLAN filter can use a MAC filter as a sub-recipe, but a match of the MAC should not be considered as an action. To indicate this, the hardware sets, for root entries added only as part of another recipe with a *Valid* = 0 bit. This entry is ignored as part of the rules resolution. The only exception is for recipes for which the *allow_pass_L2* is set. In this case, an action with *Valid* = 0 is considered as a match.

Caution: If it is important to differentiate between an entry added as a part of a larger lookup and an entry added to allow L2 filtering. The recipe used with *allow_pass_L2* set should not be used as part of other recipes.

7.8.5.3.5 Forwarding Actions

The following forwarding actions are available:

- **Forward to a VSI** — Using VSI forwarding action with *LIST* bit cleared.
- **Forward to a list of VSIs** — Using VSI forwarding action with *LIST* bit set.
- **Forward to a queue group within a VSI** — Using ToQueue action with non zero Qregion. The region base equals "Queue Index" and the region size equals 2^(Qregion). The priority used is defined by the *Q_PRI* bit.
- **Forward to a queue within a VSI** — Using ToQueue action with zero Qregion.

  Note: The forwarding actions to a queue do not include the VSI as part of the command and uses the absolute queue. The forwarding action to a queue group uses the absolute queue number of the first queue in the queue group. The VSI number is extracted from the queue number using a queue to VSI table described in Section 7.8.8.3.
- **Do not forward to the LAN** — Using the *LANEN* field. If *LANEN* bit is cleared in any of the matched lookup with a forwarding or pruning action, the packet is not sent to the LAN. This action is relevant only for transmit packets.

  Note: This action is forced if *LANENABLE* field in the source VSI context is cleared. This means that if *LANENABLE* is cleared, the packet is not sent to the LAN independent of the programmed actions.
- **Allow loopback** — Using the *LBEN* field. If *LBEN* bit is set in any of the matched lookup with a forwarding action, the packet is loopback.

  Note: This action is gated by the *ALLOWLOOPBACK* field in the source VSI context. This means that if ALLOWLOOPBACK is cleared, the packet is not sent to the loopback path independent of the programmed actions.
- **Drop packet**

  See below for details on how multiple forwarding actions are applied.

7.8.5.3.5.1 Forwarding Actions Priorities

Each recipe contains two priority values used by forwarding actions:

- **Priority** — A number between 0-7 (represented in the recipe as a one hot bitmap).
- **Join Priority** — An 8-bit bitmap.

The following algorithm is used to define which forwarding actions are retained to create the ultimate forwarding action:

```
For all the matching recipes resulting in a "forwarding" and "to queue" actions {
    find maximum value of Priority --> max_priority
}
For all the matching recipes resulting in a "forwarding" and "to queue" actions {
    If (join_priority[max_priority] == 1) {
    // note that the bit of the max priority should be set in the join priority vector
        Add VSI/VSI list/Queue to resulting VSI list
    }
}
```

7.8.5.3.6 Event-Based Mirroring Action

This action allows event-based mirroring of the traffic for which the action is enabled to a mirror VSI. Event based mirroring is independent of the forwarding priorities and the mirror VSI is always added even if the packet is dropped.

Event mirroring action can cause a loopback, even if loopback is not enabled for the ingress VSI.

Port-based mirroring is also available and is described in Section 7.8.5.4.1.

Note: The mirror VSI should be used only to receive mirror traffic (either event-based or port-based).

7.8.5.3.7 Prune Action

The prune action acts on the VSI list created by the forwarding action. It can either:

- Remove VSI listed in a VSI list from the forwarding list (AND NOT prune - PruneT = 1)
- Remove VSI not listed in a VSI list from the forwarding list (AND prune - PruneT = 0)

Each pruning action is enabled for Rx/loopback or Tx traffic using the Egress (Rx) pruning enables and Ingress (Tx) pruning enables fields in the Add VSI command. The enable index to use is set in the *Prune Index* field in the recipe pointing to this action.

This action is additive and not dependent on the priority of the recipe. Multiple prune actions can be applied.

In addition to these actions, there can be anti-spoofing rules and source pruning rules that also generate an implicit pruning. These rules are implemented in the default configuration using recipes with an inverse action bit set as described in Section 7.8.5.1.1.

7.8.5.3.8 Generic Value Action

This action allows setting of metadata forwarded to the next stages of the pipe. The Generic Offset, Generic Value, and Generic Priority define the location, the content, and the priority of the data in the metadata bus.

The encoding of the generic offset is as follows:

Table 7-15. Generic Offset Encoding

Value	Field	Priority Field	Notes
0x0	genericMD_word_0	genericMD_word_0_prio	
0x1	genericMD_word_1	genericMD_word_1_prio	
0x2	genericMD_word_2	genericMD_word_2_prio	
0x3	genericMD_word_3	genericMD_word_3_prio	
0x4	genericMD_word_4	genericMD_word_4_prio	
0x5	flowID_word_0	flowID_prio	The priority is common to these two fields. If both are set through actions, the priority is taken from the flowID_word_0 action.
0x6	flowID_word_1		
0x7	RX_descr_structure_prof_idx	RX_descr_structure_prof_idx_prio	Only first six bits of the word are set.

7.8.5.3.9 Statistics Actions

Statistics gathering can be done at four different levels:

- At the switch ID level:
  - Per switch ID packets are counted that matched a switch ID, but were dropped.
  - If the switch ID is part of a VEB. There are 32 VEB statistic blocks. In this case, statistics are gathered at the VEB level and at the VEB/UP level.
- At the VSI level, if packet is forwarded to a VSI (egress) or sent by a VSI (ingress).
- At the port level if packet is forwarded to a port (egress) or received from a port (ingress).
- Gather event statistics (usually VEB/VLAN) in statistic block N out of 128 available blocks. These counters are activated using the "statistics" action.

Note: See Section 7.8.6.1.1 for limitations on multiple event counting per packet.

Unless noted otherwise, dropped packets are not counted in the statistic counters.

7.8.5.3.9.1 Statistics Registers

The following registers are available for statistics gathered by the binary classifier:

- 256 per switch ID registers to count drops — GLSWID_RUPP[255:0].
- 32 instances of VEB, VEB/UP statistics described in Section 9.6.5.2.
- 128 instances of block statistics also described in Section 9.6.5.2.

All these counters are accessible to all the PF drivers and are read only registers that wrap around after reaching their maximum value.

7.8.5.3.9.2 Association of VEB Statistic Block to a VEB

The following flow can be used to associate a VEB statistic block to a VEB.

1. When creating a VEB, decide if a counter is needed.
2. If yes, allocate a VEB counter by using the Allocate Resources (0x0208) command with a resource type of VEB statistic counter (0x0) and get a counter ID.
3. For each VSI added/updated to this VEB, in the Add VSI command (0x0210) set *VEB_CNT_VAL* = 1 and *VEB_CNT_ID* to the associated counter ID.

7.8.5.3.10 VSI Bitmaps

Three types of actions can point to list of VSIs:

- Forward to a list of VSIs
- Prune (AND)
- Prune (AND NOT)

The E810 provides two set of 1024 per-VSI bitmaps that allows setting all the VSIs that are to be included in the action. The first set can be pointed by forwarding actions. The second set can be pointed by the two types of pruning actions.

The 1024 VSI bitmaps used for pruning actions are divided to four banks of 256 entries each. To achieve the nominal performance, pruning actions applied to a single packet should be divided between the four banks, so that in each bank there are up to two pruning lists.

7.8.5.4 Post Processing Actions

In addition to the actions generated by the lookup table, a few actions are generated by dedicated logic:

- Transmit Packets received with a non-zero *SWTCH* field from a VSI where the VSI_SRCSWCTRL.*ALLOWDESTOVERRIDE* bit is set in their descriptor are forwarded according to this flag:
  - If *SWTCH* = 01b, the packet is sent to the LAN.
  - If *SWTCH* = 10b, the LAN is removed from the list of destinations.
  - If *SWTCH* = 11b, the packet is sent to the VSI specified in the *TARGET_VSI* field in the descriptor. In this case, the switch cannot set a specific queue destination within the VSI.

    Note: If the VSI_SRCSWCTRL.*ALLOWDESTOVERRIDE* bit is not set, and a non-zero value is set, the packet is dropped. In this case, the GL_OVERRIDEC.*OVERRIDE_ATTEMPTC* field is increased, and the GL_OVERRIDEC.*LAST_VSI* field is set with the value of the source VSI.

    If *SWTCH* = 01b or 11b, the switch is bypassed and no action is applied to this packet (for example, event counters, generic actions, prune, MAC source pruning).

    If *SWTCH* not equal zero, the ingress prune rules do not apply.

    Note: When *SWTCH* = 11b, the switch does not check that the packet is sent to a VSI within the same PF. Use with care.
- SWPE — A packet received with this bit set, the packet is sent back to the source VSI.

- Store Bad Packets — If PRT_SBPVSI.*SBP* is set, packets received with MAC errors as defined in Section 3.2.1.8 are forwarded to queue 0 of the VSI defined in the Bad Frames VSI (PRT_SBPVSI.*BAD_FRAMES_VSI*).
- Default action for transmit packets — A transmit packet is sent by default to the LAN, unless a "do not forward to LAN" action is set.

7.8.5.4.1 Port-Based Mirroring

The E810 supports 64 port-based mirror rules that can be a combination of the following four types. These types are implemented using dedicated hardware and not through the regular actions:

- **Mirror ingress traffic from host** — Mirror packets sent from a specific VSI.
- **Mirror egress traffic to host** — Mirror packets sent to a specific VSI.
- **Mirror ingress traffic from LAN** — Mirror packets received from a specific LAN port. This includes packets with L2 errors.
- **Mirror egress traffic to LAN** — Mirror packets sent to a specific LAN port.

Each ingress or egress VSI/Port can be mirrored by one rule only.

Per mirror rule, a mirror VSI is defined that specifies where packets are mirrored. There is no support for mirroring to an external port.

To differentiate packets received according to the different rules, the matched rule is indicated in the Mirror Rule ID (MIRR) field in the receive descriptor.

In addition to these rules, specific traffic can be mirrored by assigning a mirror action to a specific lookup. In this case, the mirrored packet is not identified with a Mirror Rule ID (MIRR) in the receive descriptor. Event based mirroring is described in Section 7.8.5.3.6.

Note: The mirror VSI should be used only to receive mirror traffic (either event-based or port-based).

Packets that are dropped by the switch can still be mirrored due to ingress mirror rules. In this case, the packet is counted as dropped in the GLSWID_RUPP counters and is not counted in the regular switch counters.

7.8.6 Egress Post Processing Actions

7.8.6.1 Actions Merging and Priorities

As described above, each recipe can provide one or more actions. Each recipe is associated with a priority and a join priority bitmap applied to forwarding actions only.

The following rules describe the resulting actions in decreasing order of priority:

- If drop action exists, drop packet.
- If any filter matched a non-default action (including a non-valid action) and has the Pass L2 filter in its recipe, the packet is assumed to pass L2 filter. If pass L2 filter is false, the subsequent forwarding actions merging step should take into account only actions not requiring L2 filtering. Otherwise, all actions are considered.

- Forwarding actions merging:
  - From all recipes with forwarding actions, devise the highest priority (H-P) available.
  - Find forwarding actions with H-P bit set in join priority bitmap in recipe and merge VSI destination list of all these actions. Following is an example of this algorithm:

Recipe	Priority	Join Priority	Forwarding Action
1	4	00010000	VSI_1
2	3	00011000	VSI_2
3	2	00001100	VSI_3
4	2	00011100	VSI_4
Result	H-P (4)		VSI_1, VSI_2, VSI_4

  - If two of the selected actions points to the same VSI, but one of them defines a queue, the packet is sent to this queue.
  - If no "allow loopback" action found and packet is Tx, remove from list all local VSIs.
  - If "do not forward to LAN" action found and packet is Tx, remove "LAN" from list.
  - If the manageability filters indicate the packet as exclusive to the BMC (MNGONLY), modify the list to include only the BMC VSI (including removal of forwarding to LAN for OS2BMC traffic).
- Pruning actions:
  - All pruning actions should be applied after the forwarding list is ready.

    Note: Currently there are two pruning actions applied on the list: Egress/Ingress VLAN (AND), and Source MAC Address (AND NOT). Additional actions (source VSI pruning) are applied per egress replica. See Section 7.8.6.5 for details.
  - Add ingress mirroring rules.

    Note: Egress mirroring rules are applied at a later stage, when packets are replicated.
- Statistic actions are always applied if they exist.

7.8.6.1.1 Actions Limitations

For a single packet lookup the following limitations applies:

- At most, 16 actions can be generated. The combined VSI forwarding actions counts as a single action.
- Per egress VSI, a single destination Queue can be specified. If more than one is specified, the result is unexpected.
- For a all replications, up to two destination queues for different VSIs can be specified. If there are more than two actions defining a queue, then the first two "to queue" actions from the recipes with the highest RIDs are retained, and all other are translated to simple VSI forwarding actions.
- Per profile, up to 32 lookup lines can be selected and 24 of them can be recipe roots.
- If more than one hit action points to the same counter for the same packet, it is counted only once. The counter with the higher recipe index is counted. If the same wide action has several statistic actions, the last action is chosen.

7.8.6.2 Bypassing Actions

The following flags in metadata result in a bypass of the actions described above:

- If the *Switch_directives*[1:0] field (from descriptor) is set to 01b (to LAN), Tx packet is sent to LAN only (not relevant for receive packets).
- If the *Switch_directives*[1:0] field (from descriptor) is set to 11b (specific VSI), Tx packet is sent to the VSI set in the *target_vsi* only (not relevant for packets received from LAN).
- If *SWPE* bit is set, the packet is sent back to the source VSI.
- If *l2_mac_err* bit is set and pass bad frame is allowed for the receive port (PRT_SBPVSI.*SBP*), the packet is sent to Bad Frames VSI (PRT_SBPVSI.*BAD_FRAMES_VSI*).
- If *pkt_is_marker* bit is set, no action is taken.
- If *hit_mirror* bit is set, the packet is looped back only for mirroring, so other lookups should not be applied.
- If *target_vsi_is_mng* bit is set, the packet is looped back only for forwarding to manageability through MDEF filters, so other lookups should not be applied.
- If storm control is active, and packet should be dropped (broadcast/multicast), apply storm control actions and drop the packet. See Section 7.8.6.3 for details.

7.8.6.3 Storm Control

As there is no separate path for multicast and broadcast packets, too many replicated packets might cause congestions in the data path. To avoid such scenarios, broadcast and multicast storm control rate limiters are added. The rate controllers define windows and the maximal allowed number of multicast or broadcast bytes/packets per window. Once the threshold is crossed, different types of policies can be applied.

7.8.6.3.1 Storm Control Functionality

Note: Most of the parameters in this section refer to the Set Storm Control Configuration admin command described is Section 7.8.12.4.1.

The time interval over which Broadcast Storm Control is performed is controlled by three factors:

- PRT_SWT_SCBI register (loaded from NVM)
- Port speed
- INTERVAL value (see Section 7.8.12.4.1)

The first two factors determine the Unit time interval as described in Table 7-16. The interval is automatically chosen internal to hardware based on port speed. The third factor (*Interval* field) determines how many of such unit intervals are considered for one Storm Control Interval (SCI).

Table 7-16. Storm Control Basic Interval by Speed

Port Speed	MIN Time Arrival	MAX Time Arrival	Default Time Interval (1 Mbit)
100 Gb/s	1 µs	1 ms	10 µs
50 Gb/s	2 µs	2 ms	20 µs
25 Gb/s	4 µs	4 ms	40 µs
10 Gb/s	10 µs	10 ms	100 µs
1 Gb/s	100 µs	100 ms	1 ms
100 Mb/s	1 ms	1 second	10 ms

The number of 64-byte chunk of broadcast or multicast packets that are allowed in a given SCI is determined by the Set Storm Control admin command.

In each SCI, the E810 counts the number of relevant 64-byte chunks received. The count is done per-port and separately for broadcast and multicast packets. This includes packets received from the network and loopback packets sent by local VSI. Once one of the counters crosses the threshold defined in the admin command, a storm event is detected.

Note: Every time a threshold is updated, the relevant counter is zeroed.

The E810 supports two modes of reactions to storm events:

- Block all multicast or broadcast packets from the moment the threshold is crossed until the end of the interval. The block is removed at the end of the interval until the threshold is crossed again. This mode is set by setting the *MDICW* (for multicast) and *BDICW* (for broadcasts) bits in the PRT_SWT_SCCRL register. This mode is used as a rate limiter.
- Block all multicast or broadcast packets from the moment the threshold is crossed until a full interval without threshold crossing is registered. This mode is set by asserting *MDICW* and *MDIPW* (for multicast) and *BDICW* and *BDIPW* (for broadcasts) bits in the PRT_SWT_SCCRL register. This mode is used for storm blocking.

Any change in the storm control state (block or pass of multicast or broadcast packets) is indicated to the PF associated with this port via the PFINT_OICR.*STORM_DETECT* interrupt cause. The current state is reflected in the PRT_SCSTS register.

The number of dropped packets due to storm control events is reflected in the GLPRT_STDC[7..0] registers.

7.8.6.3.2 Storm Control Programming

The following admin commands described are used to program the storm control elements:

- Set Storm Control Configuration (see Section 7.8.12.4.1)
- Get Storm Control Configuration (see Section 7.8.12.4.2)

7.8.6.4 Packets Replication

If the combination of actions yields a list of destination VSIs, the packet is replicated to each VSI.

The replicated packets cannot be sent contiguously. If traffic from other traffic classes exists, each copy of the packet is sent when the round-robin arbiter reaches the traffic class queue. See Section 8.2.1 for a full description of the receive path QoS scheme.

7.8.6.5 Post-Replication Actions

- **Egress Mirroring** — If an egress mirror rule applies to the destination VSI, the mirroring port should be added to the destination list of the packet. Egress mirroring is programmed through the VSI_SWT_MIREG CSRs (see Section 7.8.5.4.1 for details).

  Note: A packet can be replicated multiple times to the same VSI if additional replications are added as part of port based mirroring actions, or if the VSI is modified in subsequent blocks in the pipe.

- **Source Pruning** — If the destination VSI of this replica is the same as the source VSI, and VSI_RXSWCTRL.*MACVSIPRUNEENABLE* is set for this VSI, this replica is dropped.

7.8.7 Manageability Filtering

The manageability filters are described in Section 12.4.

The manageability sub-block has its own field extractor.

The manageability traffic is directed to one of eight MDEF sets according to the port or virtual port number.

The GL_SWT_PRT2MDEF registers define the mapping from port to MDEF set. The selection between port and virtual port is done according to GL_SWT_SWIDFVIDX.*PORT_TYPE* field.

The field extractor for all the profiles is pre-programmed from NVM and should include extraction of the following fields which are part of the input set needed for manageability filtering:

- Destination MAC Address
- Destination IP Address (including ARP target IP)
- Destination and Source TCP/UDP port
- EtherType
- ICMP (or IP protocol)
- ICMPv6 types
  - 0x86 (134d) — Router Advertisement.
  - 0x87 (135d) — Neighbor Solicitation.
  - 0x88 (136d) — Neighbor Advertisement.
  - 0x89 (137d) — Redirect.

• Flex filter match (in metadata)

The result of the manageability filters is a forwarding decision for the traffic of the BMC. The used VSI is defined in the per decision filter, per MDEF set PRT_MNG_MDEFVSI registers. In case of multiple matches, the VSI assigned to the MDEF with the highest index is used.

7.8.7.1 Manageability Configuration

The field vector used for manageability filtering is described in Table 7-17 and is configured using the regular field extractor logic programming flow.

Table 7-17. Default Field Vector for Manageability Filtering

Filter	Words	Notes
Destination MAC Address	0-2	Single/outer
Outer VLAN	3	2nd in case of double VLAN
EtherType	4	Outer
ARP	5	Operation field
	6-7	TPA
Neighbor Discovery and MLD	8	Type field of ICMPv6
RMCP/Flexible Port - Dest	9	TCP destination port
	10	UDP destination port
ICMP	11	Protocol field of single/outer IPv4
IP Address	12-13	Single/outer IPv4
	14-21	Single/outer IPv6
Flexible - SRC	22	TCP Source port
	23	UDP Source Port
Non Valid Packets — If any of these word's valid bit is set, the packet is not a candidate for MDEF lookup. Currently includes outer IPv4 and IPv6.	24	Inner IPv4 - LSB
	25	Inner IPv6 - LSB
	26-31	Reserved

In addition, a set of masks is applied to the 64-bit flag bus to define if a packet is candidate to the MDEF filters. These masks filters packets like GRE or MPLS, which are not relevant for manageability traffic.

7.8.8 Virtual Station Interfaces

Virtual Station Interfaces (VSIs) are the connections from the switch to entities interfacing with the host. It can be either the entire queue set of a PCIe function, or part of the queues of a function.

There can be up to 768 VSIs in the E810. All the VSIs are equivalent.

At init time, each physical function is assigned a VSI.

Other VSIs can be assigned to the physical function or to Virtual functions and can be used for the following purposes:

- VMDq2
- VMDq1
- Control ports
- iWARP traffic
- Mirroring

In addition, EMP VSIs can also be defined. The EMP VSIs are used for:

- Pass-through traffic
- Control ports

Each VSI has an Egress and Ingress context as described in Section 7.8.8.1 and Section 7.8.8.2, respectively. In addition, a queue context mapping queues to VSIs is also available.

Note: Additional VSI contexts are part of the ACL and Filtering blocks.

7.8.8.1 Egress VSI Context

A per-VSI table containing the context of the VSI with the following parameters looked up with the destination VSI (VSI_RXSWCTRL and VSI_VSI2F registers):

- VSI_VSI2F.*VFVMNUMBER*, VSI_VSI2F.*PFNUMBER*, VSI_VSI2F.*FUNCTIONTYPE* — PF/VF to which the VSI belongs.
- VSI_VSI2F.*BUFFERNUMBER* — The buffer in the manageability block to which traffic is sent if *FUNCTIONTYPE* = EMP.
- VSI_RXSWCTRL.*MACVSIPRUNEENABLE* — Defines if the VSI can receive packets with its own MAC Address. If set, such packets are dropped. Enabling recipes where the *inverse_action* is set for Rx packets.
- VSI_RXSWCTRL.*SRCPRUNEENABLE* — Defines if a VSI can receive packets it sent.
- VSI_RXSWCTRL.*PRUNEENABLE*[3:0] — Enables prune actions for Rx traffic. According to the *prune_index* in recipe
- VSI_VSI2F.*VSI_ENABLE* - VSI Enable.

7.8.8.2 Ingress VSI Context

A per-VSI table containing the context of the VSI with the following parameters looked up with the source VSI (VSI_SRCSWCTRL registers):

- *ALLOWDESTOVERRIDE* — Defines if a source VSI is allowed to bypass the switch using the *SWTCH* flag in Tx-Descriptor.
- *ALLOWLOOPBACK* — Defines if packets sent from a VSI can be looped back.
- *LANENABLE* — Defines if packets sent from a VSI can be sent to the LAN.
- *PRUNEENABLE*[3:0] — Enables prune actions for Tx traffic. According to the *prune_index* in recipe.
- *MACAS* — Allows MAC anti-spoof for this VSI. Enabling recipes where the *inverse_action* is set for Tx packets. If set, a packet is sent only if at least one recipe with inverse action provides an indication of a matching VSI to the source VSI.

7.8.8.3 Queue Context

A 2K-entries table mapping absolute queue numbers to VSIs and relative queue numbers.

All tables are filled by the Add VSI command.

7.8.9 Classifier Performance

The classifier is able to classify up to 171 Mpps Rx/Tx packets assuming up to 10 lookups per packet. The classification includes a forwarding decision and two ingress pruning decisions. Other actions, like statistics or storm control, should not impact the performance of the classifier. Replication or mirroring of a packet causes a reduction of the number of packets handled per second.

The classifier supports up to 500 VM configurations per second (includes VSI and filters configurations) in run time, and up to 5000 filter configurations at init time, to allow bring-up of a complex topology in a short time. This assumes bundling multiple filters configurations within a single Add Switch Rules command (Section 7.8.12.6.1).

There is no requirement on the profile configuration performance.

7.8.10 Resource Allocation

The following resources, including Switch, Flow Director, RSS, and per-pipe stage profile configuration resources, are managed by the firmware:

Table 7-18. Resource Types

Resource ID	Resource Type	Allocation Type	Notes	Pre-Allocatable in NVM
0x0	VEB Statistic Counter	Specific entries	Up to 32 VEB counter blocks.	
0x1	Event Statistic Counter	Specific entries	Up to 128 events (for example, VEB/VLAN) counter blocks.	
0x2	Mirror rule	Specific entries	Up to 64 mirror rules.	
0x3	VSI List - Replication	Specific entries	Up to 1K lists.	
0x4	VSI List - Prune	Specific entries	Up to 1K lists	
0x5	Recipe	Specific entries	Up to 64 recipes. At init time, all recipes are shared. The default recipes are shared and available for all the functions. If a global package is loaded, all the recipes are replaced by the recipes of the package, and are shared and available for all the functions.	
0x6	Switch Profiles	Specific entries	Up to 256 profiles.	
0x7	SWID	Specific entries	Up to 256 switch IDs.	
0x8	VSI	Specific entries	Up to 768 VSIs.	Yes
0x9	FLU Entry	Quantity only	Up to 32K FLU entries. The allocation and de-allocation assumes each filter uses new entries, so the actual usage can be lower than what is reflected by this command.	Yes
0xA	Wide Table 1	Specific entries	Total 8K wide actions. See Table 7-13 for the distribution.	
0xB	Wide Table 2	Specific entries		
0xC	Wide Table 4	Specific entries		
0xD-0x1F	Reserved for future switch resources	N/A		
0x20	Global RSS Hash	Specific entries	Up to 16 RSS hash tables.	
0x21	Flow Director Counter Block	Specific 256 counters blocks	One of 32 blocks (total 8K counters).	

Table 7-18. Resource Types [continued]

Resource ID	Resource Type	Allocation Type	Notes	Pre-Allocatable in NVM
0x22	Flow Director Guaranteed Entries (for software-based filters)	Quantity only	The total number of available Flow Director guaranteed entries is GLQF_FD_SIZE.*FD_GSIZE*. To increase this resource, use the Allocate Resources AQ command. To decrease this resource, use the Free Resources AQ command. Using the Get Allocated Resource Descriptors AQ command for this resource is not relevant and returns an empty buffer.	
0x23	Flow Director Shared Entries (for ATR filters)	Quantity only	The total number of available Flow Director guaranteed entries is GLQF_FD_SIZE.*FD_BSIZE*, but there can be over-subscription between PFs. To increase this resource, used the Allocate Resources AQ command. To decrease this resource, use the Free Resources AQ command. Using the Get Allocated Resource Descriptors AQ command for this resource is not relevant and returns an empty buffer.	
0x24-0x2F	Reserved for future filtering resources	N/A		
0x30	Flexible Descriptors Programming	Specific entries	Up to 64. Shared by default. These resources are not cleared by PFR. ***Note:*** Resources #0, #1, and #7 are pre-allocated, and never allocated to a specific PF. See Section 7.6.4.	
0x31-0x3F	Reserved for filter resources	N/A		
0x40-0x47	Reserved for Parser Profile Builder	N/A		
0x48	Switch Field Vector Table	Specific entries		
0x49	Switch Profile ID TCAM Entries	Specific entries		
0x4A-0x4F	Reserved for Switch Profile Builder	N/A		
0x50	ACL Field Vector Table	Specific entries		
0x51	ACL Profile ID TCAM Entries	Specific entries		
0x52-0x57	Reserved for ACL Profile Builder	N/A		
0x58	Flow Director Field Vector Table	Specific entries		
0x59	Flow Director ID TCAM Table Entries	Specific entries		
0x5A-0x5F	Reserved for Flow Director Profile Builder	N/A		
0x60	Hash Field Vector Table	Specific entries		
0x61	Hash Profile ID TCAM Entries	Specific entries		
0x62-0x67	Reserved for Hash Profile Builder	N/A		
0x68	Quad Hash Field Vector Table	Specific entries		

Table 7-18. Resource Types [continued]

Resource ID	Resource Type	Allocation Type	Notes	Pre-Allocatable in NVM
0x69	Quad Hash Profile ID TCAM Entries	Specific entries		
0x6A-0x6F	Reserved for Quad Hash Profile Builder			
0x70-0x7F	Reserved			

The following principles are defined to support these resources management:

- Every resource could be Persistent (unmanaged) or Dedicated (exclusive). Profile builder resources can be also Shared.
- Dedicated resource rules:
  - — One resource / one owner.
  - — Dedicated resource allocation/de-allocation (explicitly by the owning driver).
  - — If a device driver attempts to reference/modify a dedicated resource owned by another function's driver, it gets an error (shared resource could be referenced/modified by any device driver).
  - — During PFR, firmware clears all the dedicated resources owned by the function.

  Note: Profile Builder resources are defined as Dedicated using the Download Package AQ command. Their ownership is kept across a PFR. However, for Profile ID TCAM, the resource's hardware information is cleared during PFR.
- Persistent resources rules:
  - — Not owned by any function driver.
  - — Allocation/de-allocation (explicitly by any driver).
  - — Any modification possible.
  - — PFR has no impact. Cleared by CORER.

  Note: Profile Builder resources can be deallocated only using the Download Package AQ command sequence.
- Shared resources rules (relevant only to Profile Builder resources):
  - — Set ownership as "shared" is done using Download Package AQ command.
  - — Can be owned by any function driver using Allocate Resource AQ command (and freed using the Free Resource AQ command).
  - — Modification is possible using the Update Package AQ command.
  - — PFR clears PF usage of resource. For Profile ID TCAM, the resource's hardware information is cleared when no PF is using the resource.
- If the software device driver attempts to reference/modify unallocated resource, it gets an error.

The commands used to manage the resources are described in Section 7.8.12.2.

Profile builder ownership is defined in Table 7-192.

7.8.10.1 Initial Resource Allocations

Resources can be pre-allocated from NVM to a PF, or can be dynamically assigned using the commands in Section 7.8.12.2. The allocation in NVM is done through the SR PF Allocations NVM section. This section allows for the allocation of up to two resources per PF, by setting the "PF allocations -Type" and "PF allocations -Value" words with {resource type, number of resources allocated} pairs.

Regardless, for each PF, a VSI and an FLU entry are allocated per function (PF or VF) associated with this PF. If a different allocation of VSI or FLU entries is specified in NVM, the maximum value between these two is taken to guarantee minimal allocation per PF/VF.

7.8.11 Binary Classifier Configuration

7.8.11.1 Overview

The configuration of the binary classifier can be done in three modes:

- Basic configuration loaded from NVM. This is a basic configuration that provides a basic L2 switching configuration. This configuration can never be removed.
- Public configurations loaded by driver that can be used by other drivers on the device. For example, a driver loading a set of filters to match the encapsulation used in the network (NVGRE, Geneve, and so on), that can be used by other drivers connected to the same network. Such a configuration can be removed only if no driver is registered to use it.
- A private configuration loaded by a driver and not exposed to other drivers. The driver is free to remove this type of configuration independently.

7.8.11.2 Parameters Summary

Table 7-19 summarize the parameters configurable in the switch and the way to configure them.

Table 7-19. Configurable Parameters

Parameter	Config Source	Mechanism	Rate	Section Reference
Profile Selection Parameters	NVM	CSRs	Once	7.8.11.3.1 through 7.8.11.3.3
Initial Recipes and Default Profile	NVM	CSRs	Once	
Initial Population (Basic L2 Filtering)	Firmware	See lookup population.	N/A	
New Control Domain - Add/Remove	Software via firmware	CSRs	500/s	
New Profile - Add/ Remove	Software via firmware	CSRs	1000/s. Up to 5000/s combined population requests	
New Set of Recipe - Add/remove	Software via firmware	CSRs	1000/s	
(Dis)Association of Traffic to Profile	Software via firmware	CSRs	N/A	

Table 7-19. Configurable Parameters [continued]

Parameter	Config Source	Mechanism	Rate	Section Reference
Add/remove VSI	Software via firmware	CSRs		
Lookup Population	Software via firmware (using packet header) Each population request in a different AQ command.	Command FIFO (lookup and action). CSRs (VSI bitmaps and large actions).		
Lookup Removal	Software via firmware (using index)	Command FIFO (lookup and action) CSRs (VSI bitmaps and large actions)		
FLR/PFR, VFR	Firmware	Hardware iterators to get lookup. Lookup removal as above.		

7.8.11.3 NVM Loaded POR

7.8.11.3.1 Default Profiles

Four profiles are defined from NVM:

- **Bypass profile (0x0)** — This profile is used for packets that are not impacted by the switch block.
- **Default profile (0x1)** — Includes all traffic not included in any other profiles.
- **Control Packets profile (0x2)** — Includes LLDP packets and flow control packets (PTYPEs = 6, 278).
- **Hardware handled packets (0x3)** — This profile is used for packets for which there is no lookup and special hardware is used to implement the decision.

Note: All the flags that are part of the bypass and hardware handled profiles have dedicated hardware to handle them. The inclusion in these profiles is to save the need for regular recipe lookup.

7.8.11.3.2 Default Field Vector

The default field vector is generated for the Default Profile (1) and Control Packets Profile (2), and contains the following data:

Table 7-20. Default Field Vector

Offset (Words)	Content	Notes
0	Switch ID	Mandatory in all extractions.
1	MAC DA 0	
2	MAC DA 1	
3	MAC DA 2	
4	MAC SA 0	
5	MAC SA 1	
6	MAC SA 2	
7	Outer VLAN	Not used by basic filters. Equals zero if not present.

Table 7-20. Default Field Vector [continued]

Offset (Words)	Content	Notes
8	VLAN	Equals zero if not present
9	EtherType	

For the Bypass Profile (0) and Hardware-Based Profile (3), the extracted vector is empty.

7.8.11.3.3 Default Recipes

The first 11 filters defined in Table 7-10 are loaded from NVM.

7.8.11.4 Resets

The entire switch configuration is cleared and reloaded from NVM upon a CORER or higher reset.

At a PFR or FLR reset, all the entries populated by the PF, all the recipes and profiles private to this PF, and all the VSI context for VSIs allocated to this PF are cleared.

At a VFR or VMR, there is no modification of the switch configuration.

7.8.12 Software Programming Model

The programming of the different switching elements is done using admin commands. Commands to configure a switching element can be received only from the control port of the element.

7.8.12.1 Switch Configuration Admin Commands Summary

The set of commands is based on X710/XXV710/XL710 switch commands with adaptations to the new programming paradigm of the E810.

Table 7-21 summarize the different commands used to configure the binary classifier.

Table 7-21. Switch Configuration Admin Commands (0x02xx)

Command	Opcode	Description	Section Reference
Common Commands The commands common to the entire receive control pipe are described in Section 7.11.			
Generic Commands/Resource Management (0x020x)			
Get Switch Configuration	0x0200		7.8.12.2.1
Set Port Parameters	0x0203		7.8.12.2.2
Get Resource Allocation	0x0204		7.8.12.2.3
Allocate Resources	0x0208		7.8.12.2.4
Free Resources	0x0209		7.8.12.2.5
Get Allocated Resources Descriptor	0x020A		7.8.12.2.6
Other commands in this section supported in previous products are obsolete in the E810.			

Table 7-21. Switch Configuration Admin Commands (0x02xx) [continued]

Command	Opcode	Description	Section Reference
VSI Commands (0x021x)			
Add VSI	0x0210		7.8.12.3.1
Update VSI	0x0211		7.8.12.3.2
Get VSI Parameters	0x0212		7.8.12.3.3
Free VSI	0x0213		7.8.12.3.4
Port Virtualizer Control (0x022x) All the commands in this section supported in previous products are obsolete in the E810.			
VEB/Port Aggregator Control (0x023x) While these commands are no longer supported in the E810, the functionality of the VEB configuration is replaced by the SWID configuration, also knows as the Switch ID provided in the Get Switch Config and Add/Update VSI commands.			
Switch Connectivity Configuration (0x024x) All the commands in this section supported in previous products are obsolete in the E810.			
Forwarding Table Configuration (0x025x) All the commands in this section supported in previous products are obsolete in the E810.			
Storm Control Commands (0x028x)			
Set Storm Control Configuration	0x0280		7.8.12.4.1
Get Storm Control Configuration	0x0281		7.8.12.4.2
Binary Classifier Configuration (0x029x) See Section 7.11 for profiles configuration			
Add Recipe	0x0290		7.8.12.5.1
Set Recipes-to-Profile Association	0x0291		7.8.12.5.2
Get Recipe	0x0292		7.8.12.5.3
Get Recipes-to-Profile Association	0x0293		7.8.12.5.4
Binary Classifier Population (0x02Ax)			
Add Switch Rules	0x02A0		7.8.12.6.1
Update Switch Rules	0x02A1		7.8.12.6.2
Remove Switch Rules	0x02A2		7.8.12.6.3
Get Switch Rules	0x02A3		7.8.12.6.4
Clear PF Configuration	0x02A4		7.8.12.6.5
Mirroring Configuration (0x026x)			
Add Mirror Rule	0x0260		7.8.12.7.1
Delete Mirror Rule	0x0261		7.8.12.7.2

7.8.12.2 Generic Commands (0x020x)

7.8.12.2.1 Get Switch Configuration (0x0200)

After receiving this command, firmware returns a list of currently allocated and used resources (VSIs and SWIDs), so a topology tree could be built.

Table 7-22. Get Switch Configuration Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0200	Command opcode.
Datalen	4-5	(2-2048)	Length of response buffer.
Return Value/VFID	6-7		Return value. Zeroed by device driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Command Flags	16-17		Reserved.
First Descriptor	18-19	0x0	If not zero, start the report from the requested descriptor.
Reserved	20-23		Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

Table 7-23. Get Switch Configuration Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0200	Command opcode.
Datalen	4-5		Length of response buffer.
Return Value/VFID	6-7		Return value. Zeroed by device driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Flags	16-17		Copy of request flags.
New Element	18-19		If not zero, indicates that not all the configuration was returned and a new command should be sent with this value in the *First Descriptor* field.
Number of Elements	20-21		Number of elements in response buffer.
Reserved	22-23		Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

Table 7-24. Get Switch Configuration Response Buffer

Offset (Bytes)	Description
0 -5	Port/VSI Element 1 (see Table 7-25).
…	
6*(n-1):6*n-1	VSI Element #n.

Table 7-25. Get Switch Configuration – Port/VSI Element

Offset (Bytes)	Description
0-1	**Bits 9:0:** VSI number/Port Number **Bits 13:10:** Reserved **Bits 15:14:** 00b = Physical Port 01b = Virtual Port 10b = VSI 11b = Reserved
2-3	SWID VSI/port belongs to
4-5	**Bits 14:0:** PF/VF number, VSI belongs to. **Bit 15:** VF indication bit

7.8.12.2.2 Set Port Parameters (0x0203)

This command is used to define the default parameters of a physical port.

Table 7-26. Set Port Parameters Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0203	Command opcode.
Datalen	4-5	0x0	Length of buffer
Return Value/VFID	6-7		Return value. Zeroed by device driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Command Flags	16-17	Bitfield	**Bit 0:** Save Bad Packets If set, packets with errors are forwarded to the bad frames VSI. **Bit 2:** Enable Double VLAN If set, this port expects double VLAN packets. ***Note:*** After setting this bit, the software device driver should associate the port to a double VLAN profile in the parser. All other bits are Reserved.
Bad Frames VSI	18-19		Defines the VSI to which bad frames are forwarded. **Bit 15:** Valid VSI - ignored in this field **Bits 14:10:** Reserved **Bits 9:0:** Number of the VSI ***Note:*** Relevant only if Command Flags.Save Bad Packets is set.

Table 7-26. Set Port Parameters Command [continued]

Name	Byte.Bit	Value	Remarks
Port Switch ID	20-21		**Bit 15:** Valid Switch ID **Bits 14:8:** Reserved **Bits 7:0:** Switch ID ***Note:*** If a switch ID is already allocated to the port, it is replaced by the new one.
Reserved	22-31	0x0	Reserved.

Table 7-27. Set Port Parameters Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0203	Command opcode.
Datalen	4-5	0x0	Length of buffer
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. The following response might be returned by this command: ENOENT = Index is invalid for one of the chosen resources. EACCES = PF does not own one of the chosen resources.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Command Flags	16-17	Bitfield	
Bad Frames VSI	18-19		
Default Switch ID	20-21		Returns the Switch ID assigned to the port.
Reserved	22-31	0x0	Reserved.

7.8.12.2.3 Get Resource Allocation (0x0204)

This command returns the number of resources allocated to a function from each type of resource. For all the subsequent resource management commands, the encoding of the resource type is described in Table 7-18.

Table 7-28. Get Resource Allocation Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0204	Command opcode.
Datalen	4-5		Length of response buffer. Should be enough to store response.
Return Value/VFID	6-7		Return value. Zeroed by device driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-23		Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

Table 7-29. Get Resource Allocation Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0204	Command opcode.
Datalen	4-5		Length of response buffer. Should be number of resource returned x resource entry size.
Return Value/VFID	6-7		Return value. Zeroed by device driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of Entries	16-17		Number of entries in buffer.
Reserved	18-23		Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

Table 7-30. Get Resource Allocation Response Buffer Entry

Offset (Bytes)	Description
0-1	**Bits 6:0:** Resource type - see above for encoding. **Bits 15:7:** Reserved
2-3	Total Capacity — Total number of resources from this type available to all functions (fixed according to hardware capabilities).
4-5	Allocated Function — Number of resources from this type allocated to this function (for most resources but hash based, this means actual entries are owned by the function).
6-7	Allocated Shared — Number of resources from this type allocated as shared.
8-9	Total Free — Total number of resources from this type still unallocated and not reserved by any function.

7.8.12.2.4 Allocate Resource (0x0208)

Table 7-31. Allocate Resource Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0208	Command opcode.
Datalen	4-5	(6-4096)	Length of command/response buffer.
Return Value/VFID	6-7		Return value. Zeroed by device driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of Entries	16-17		Number of resource entries in buffer.
Reserved	18-23		Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

In response, firmware fills buffer (See Table 7-33) with corresponding resource descriptors.

Table 7-32. Allocate Resource Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0208	Command opcode.
Datalen	4-5	(6-4096)	Length of command/response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. Codes returned by firmware: OK = Success EINVAL = One of the requested parameters is invalid. ENOSPC = Allocation failed, all resources until the problematic resource (including) contain the number of resources currently available for allocation.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of Entries	16-17		Number of resource entries in buffer.
Reserved	18-23		Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

Table 7-33. Allocate Resource Command Buffer Entry

Offset (Bytes)	Description
0-1	**Bits 6:0:** Resource type (see Table 7-18 for encoding) **Bit 7:** Shared 0b = Allocate Dedicated 1b = Allocate Shared **Bits 11:8:** VSI prune list hint A bitmap that provides a hint to the firmware from which of the four banks to allocate the VSI prune lists. VSI prune list hint is a suggestion for allocation in one of the list banks, starting from the lowest bank. If the suggestion is failed due to lack of resources than there will be a try to allocate in different bank. **Bit 12:** If this bit is set, firmware scans from the bottom (last) entry (relevant only if Bit 13 is enabled, otherwise index is given in Bytes 2-3). **Bit 13:** If this bit is set, the index values (Bytes 2-3) are be ignored in command and firmware returns in response new allocated indexes. ***Note:*** Bits 12,13 are relevant only for Profile builder Resource Allocation and reserved otherwise. **Bits 15:14:** Reserved
2-3	Number of resource descriptors to allocate. Must not be zero or larger than max available number of this resource. ***Note:*** For Profile Builder Resource Allocation, if Bit 13 in Bytes 0-1 is enabled, these bytes contain an indication of which index to begin the search if it is non-zero (it is assumed that index zero is the top of the list). For example, if the PF driver owned Profile ID TCAM entry X, it might want to start scanning for new elements at X, to ensure that the new entry was higher, that is, closer to the first entry in the table.
4-...	Placeholder for resource descriptors. Two bytes/descriptor, except FLU entries (0x9) and FD filters (0x22 and 0x23): Number of resource descriptors to allocate x 2. In case of ENOSPC, first descriptor contains number of resources currently available for allocation, and no resources within this request are allocated. Two bytes for FLU and FD filters: Ignored on success, number of resources currently available for allocation on ENOSPC. The resource descriptors returned are the indexes in the resource's table. ***Note:*** When adding Profile Builder resources, firmware checks that the resource is defined as shared (inside downloaded package) and adds PF's usage of the shared resource.

Figure 7-18 describes the relationship between the different resources reported.

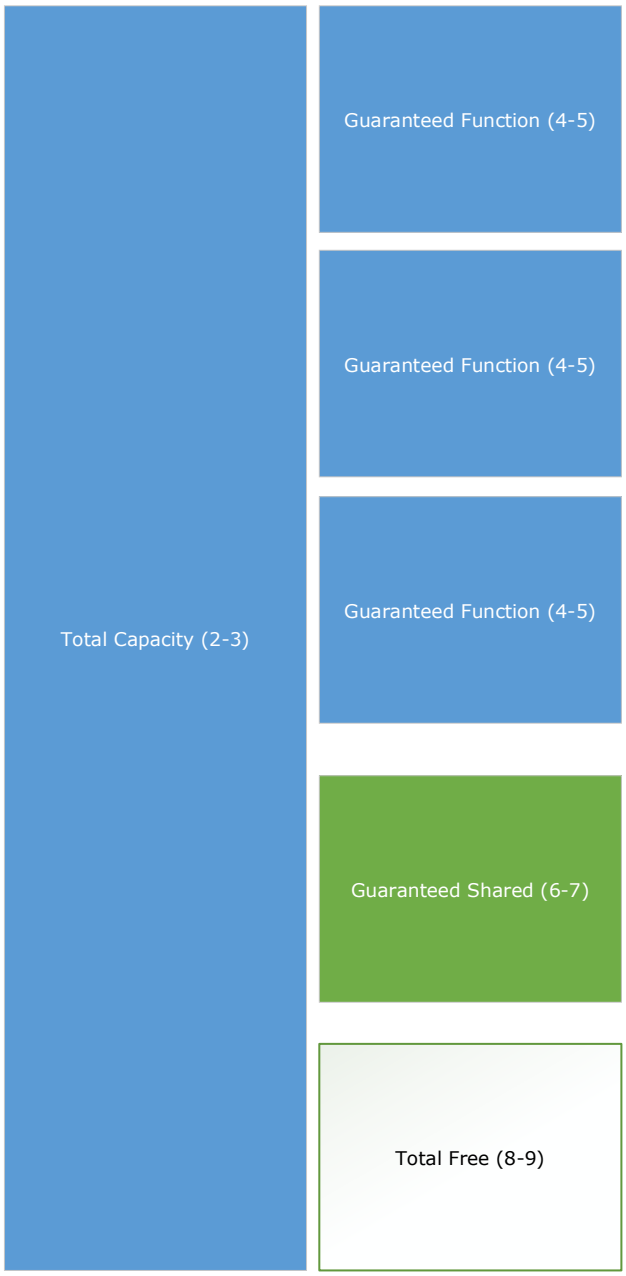


Figure 7-18. Resources Reported

7.8.12.2.5 Free Resource (0x0209)

Table 7-34. Free Resource Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0209	Command opcode.
Datalen	4-5	(6-4096)	Length of command buffer.
Return Value/VFID	6-7		Return value. Zeroed by device driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of Entries	16-17		Number of resource entries in buffer.
Reserved	18-23		Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

Table 7-35. Free Resource Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0209	Command opcode.
Datalen	4-5	(6-4096)	Length of command buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. Codes, returned by firmware: OK = Success EINVAL = Some resources could not be freed, details inside buffer. EBUSY = Trying to release a resource which is in use.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of Entries	16-17		Number of resource entries in buffer.
Reserved	18-23		Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

Table 7-36. Free Resource Command Buffer Entry

Offset (Bytes)	Description
0-1	**Bits 6:0:** Resource type (see Table 7-18 for encoding) **Bits 15:7:** Reserved
2-3	Number of resource descriptors to free. Must not be zero or larger than number of this resource allocated to this function.
4-...	Resource descriptors. Two bytes/descriptor, except FLU. In case of EINVAL, every descriptor firmware failed to released is overwritten as follows: Bit 15 is set, the rest of the bits - extended firmware diagnostic codes (like access denied, resource still referenced, and so on). Two bytes for FLU. Ignored on success, number of resources, currently possible to free on EINVAL. The resource descriptors are the indexes in the resource's table. ***Note:*** When releasing Profile Builder resources, firmware checks that the resource is defined as shared (inside downloaded package) and removes PF's usage of the shared resource. When releasing Profile ID TCAM resources, firmware clears the resource data inside hardware tables in case no PF uses the resource.

7.8.12.2.6 Get Allocated Resource Descriptors (0x020A)

This command is used to get the list of the actual resources allocated to the PF of a given resource type. This command is N/A for FD filters (0x22 and 0x23) resource types. If used for these resources, an empty buffer is returned. For FLU entries (0x9), only root lookup entries actually used are returned.

Table 7-37. Get Allocated Resource Descriptors Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x020A	Command opcode.
Datalen	4-5	(2-2048)	Length of response buffer.
Return Value/VFID	6-7		Return value. Zeroed by device driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Resource	16-17		**Bits 6:0:** Resource Type See Table 7-18 for encoding. **Bit 7:** Shared 0b = Get Dedicated 1b = Get Shared **Bits 15:8:** Reserved
First Descriptor	18-19		If not zero, start the report from the requested descriptor.
Reserved	20-23		Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

Table 7-38. Get Allocated Resource Descriptors Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x020A	Command opcode.
Datalen	4-5		Length of response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. EINVAL = One of the requested parameters is invalid.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Resource	16-17		**Bits 6:0:** Resource Type See Table 7-18 for encoding. **Bit 7:** Shared 0b = Get Dedicated 1b = Get Shared **Bits 15:8:** Reserved
Next Descriptor	18-19		If not zero, indicates that not all the configuration was returned and a new command should be sent with this value in the *First Descriptor* field.
Number of Descriptors	20-21		Number of descriptors in buffer.
Reserved	20-23		Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

Table 7-39. Get Allocated Resources Response Buffer

Offset (Bytes)	Description
0-...	Resource descriptors. Two bytes/descriptor. The resource descriptors returned are the indexes in the resource's table.

7.8.12.3 VSI Commands (0x021x)

7.8.12.3.1 Add VSI (0x0210)

This command is used to add a new VSI.

When an Add VSI command is received, the internal firmware checks if all the requested resources are available and allocate them to the VSI.

Table 7-40 describes the structure of the command buffer for the Add VSI command.

Table 7-40. Add VSI Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0210	Command opcode.
Datalen	4-5	0x80	Length of buffer.
Return Value/VFID	6-7	0x0	Return value. Zeroed by device driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
VSI	16-17	0x0	**Bit 17.7:** VSI is valid **Bits 17.6-17.2:** Reserved **Bits 17.1-16.7:** VSI Number If VSI is valid, the VSI should be from resources allocated to this PF. Otherwise, a free VSI is allocated from the shared pool.
Reserved (Connection Type)	18	0x0	Reserved.
Reserved	19	0x0	Reserved.
VF Function Number	20		Defines the VF function to which this VSI connects. Valid only if Function Type is VF. Should be ignored if VSI type is not VF. ***Note:*** The VF number here is the absolute VF number (0-255) and not the number relative to the PF first VF.
Reserved	21		Reserved.
Command Flags	22-23		**Bits 1:0:** VSI type: 00b = VF 01b = VMDq2 (a.k.a. VM) 10b = PF 11b = EMP/MNG **Bits 15:2:** Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

Table 7-41. Add VSI Command Buffer

Category	Byte.Bit	Field	Description
Valid Sections	0-1		Defines which sections are valid in the command. **Bit 0:** Switching section is valid. **Bit 1:** Security section is valid **Bit 2:** VLAN handling section is valid. **Bit 3:** Outer tag section is valid. **Bit 4:** Ingress UP translation section is valid. **Bit 5:** Egress UP translation section is valid. **Bit 6:** Rx-Queue Mapping section is valid. If set, modified queues must be disabled. **Bit 7:** Queuing option section is valid. **Bit 8:** Outer UP mapping section is valid. **Bit 9:** Reserved. **Bit 10:** ACL section valid. **Bit 11:** Flow Director section valid. **Bit 12:** PASID section valid. **Bits 13:15:** Reserved. ***Note:*** Relevant only for Update VSI command. In Add VSI command, all the sections are assumed to be valid.
Switching	2	Switch ID	Defines the switch ID to which this VSI belongs.
	3.0-3.3	Reserved (Switch ID MSB)	Reserved.
	3.4	Reserved (Is not STag	Reserved.
	3.5	Allow Loopback	If set, traffic from this VSI can be loopback.
	3.6	Allow Local Loopback	If cleared, inverse actions are applied to Rx traffic to this VSI (used in default image for MAC source pruning).
	3.7	Apply Source VSI Pruning	If set, traffic sent from a VSI can be forwarded to the same VSI. ***Note:*** It is recommended to always set this bit, unless there is an explicit need to receive back traffic sent by the VSI.
	4.0-4.3	Egress (Rx) Pruning Enables	Defines which pruning modes are enabled for egress traffic Setting Bit 0 enables VLAN pruning.
	4.4	LAN Enable	If set, traffic from this VSI can be sent to LAN.
	4.5-4.7	Reserved	Reserved.
	5.0-5.4	VEB Statistic Block ID	Defines which VEB statistic block this VSI is associated to.
	5.5	VEB Statistic Block ID Valid	If set, the *VEB Statistic Block ID* field is valid. 0b = When this bit is cleared, this association is removed. 1b = When this bit is set (in Add VSI or Update VSI command), the switch ID associated with this VSI is assigned a VEB statistic block defined by *VEB Statistic Block ID* field. Must be set for any Add/Update VSI of VSIs associated with a VEB statistic block.
	5.6-5.7	Reserved	Reserved for future switching parameters. Must be zero.

Table 7-41. Add VSI Command Buffer [continued]

Category	Byte.Bit	Field	Description
Security	6.0	Allow Destination Override	Allow the VSI to override the switching decision and fix the destination of a transmit packet. This bit should be set only for trusted VSIs. ***Note:*** If this bit is set, the *Enable MAC Anti-Spoof* bit and the VLAN anti-spoofing (*Ingress Pruning Ena*bles[0]) should not be set.
	6.1	Reserved (Enable VLAN Anti-Spoof)	Reserved.
	6.2	Enable MAC Anti-Spoof	If set, inverse actions are applied to Tx traffic to this VSI (used in default image for MAC anti-spoof).
	6.3	Reserved	Reserved.
	6.4-6.7	Ingress (Tx) Pruning Enables	Defines which pruning modes are enabled for ingress traffic Setting Bit 0 enables VLAN anti-spoofing.
	7	Reserved	Reserved for future security parameters. Must be zero.
VLAN Handling	8-9	PVID + Default UP (16 bits)	VLAN ID to use in port-based VLAN insertion. This field is relevant only if the *Insert PVID* field is set.
	10-11	Reserved	Reserved.
	12.0-12.1	VLAN Driver Insertion Mode	This field defines if the driver is allowed/should add a VLAN tag to the packets it sends. If the *Insert PVID* field is set, this field should be set to 01b. 00b = Reserved. 01b = Admit untagged/Priority tagged only - allow only packets without VLAN or with VLAN tag = 0. 10b = Admit.1Q tagged only - Allow only packets with VLAN. 11b = Allow all packets.
	12.2	Insert PVID	Port-based VLAN insertion. This bit controls the port based insertion of VLANs. Should be set for VFs/ VMDq2 VSIs according to the VMM request. If this field is set, the *PVID + Default UP* field should be set to the port-based VLAN.
	12.3-12.4	VLAN and UP Expose Mode (Rx)	This field defines how received VLAN are handled. For non-VF VSIs, 00b or 11b should be used. For VF VSIs, the mode should be set according to the VLAN awareness of the VM and the offload requested. 00b = Show VLAN, DEI and UP in descriptor (legacy behavior). 01b = Hide VLAN, show UP and DEI (VLAN ID = 0). 10b = Hide VLAN, DEI and UP. 11b = Do nothing (leave VLAN in packet).
	12.5-12.7	Reserved	Reserved for future port VLAN parameters. Must be zero.
Ingress UP Translation	16-19	Ingress UP Translation Table	Defines the UP translation table for received packets according to the following list: **Bits 16.0-16.2:** UP set if received UP is 0. **Bits 16.3-16.5:** UP set if received UP is 1. **Bits 16.6-17.0:** UP set if received UP is 2. **Bits 17.1-17.3:** UP set if received UP is 3. **Bits 17.4-17.6:** UP set if received UP is 4. **Bits 17.7-18.1:** UP set if received UP is 5. **Bits 18.2-18:4:** UP set if received UP is 6. **Bits 18.5-18.7:** UP set if received UP is 7. **Byte 19:** Reserved. This map is used to translate the 802.1P user priority bits received in the packet to the user priority exposed to the host. Relevant only if the UP is exposed to the host (*VLAN and UP Expose Mode* not equal 10b).

Table 7-41. Add VSI Command Buffer [continued]

Category	Byte.Bit	Field	Description
Egress UP Translation	20-23	Egress UP Translation Table	Defines the UP translation table for transmit packets according to the following list: **Bits 20.0-20.2:** UP set if sent UP is 0. **Bits 20.3-20.5:** UP set if sent UP is 1. **Bits 20.6-21.0:** UP set if sent UP is 2. **Bits 21.1-21.3:** UP set if sent UP is 3. **Bits 21.4-21.6:** UP set if sent UP is 4. **Bits 21.7-22.1:** UP set if sent UP is 5. **Bits 22.2-22.4:** UP set if sent UP is 6. **Bits 22.5-22.7:** UP set if sent UP is 7. **Byte 23:** Reserved. This map is used to translate the 802.1P user priority bits sent by the host to the user priority sent to the network. ***Note:*** The resulting user priority is further translated using the per TC translation table.
Outer Tags Handling	24-25	Outer Tag	The outer tag (STag/Outer VLAN) to be inserted in transmit packets. This field is relevant only if *STag Insert Enable* field is set. This field is 16 bits and should also include the default PCP priority bits to insert in the STag.
	26.0-26.1	Outer Tag Extract Tag	This field defines how received outer tags (STag/Outer VLAN) are handled. 00b = Do nothing. 01b = Extract tag and do not insert in descriptor. 10b = Extract tag from packet and expose in descriptor. 11b = Reserved.
	26.2-26.3	Outer Tag Type	Defines the type of outer tag expected. 00b = No outer tag. 01b = 0x88A8 STag. 10b = 0x8100 VLAN. 11b = 0x9100 VLAN.
	26.4	Outer Tag Insert Enable	This bit controls the port based insertion of outer tag (STag/Outer VLAN). When this bit is set, the *Accept Tag from Host* bit should be cleared.
	26.5	Reserved	Reserved.
	26.6	Accept Tag from Host	When this bit is set, the *Outer Tag Insert Enable* bit should be cleared ***Note:*** When this bit is cleared, the device prevents insertion of the tag through the descriptor.
	26.7-27.7	Reserved	Reserved for future port outer tag parameters. Must be zero.

Table 7-41. Add VSI Command Buffer [continued]

Category	Byte.Bit	Field	Description
Rx-Queue Mapping	28.0	Mapping Method	Selects between contiguous range of queues for this VSI vs. scattered range: 0b = The VSI is assigned a contiguous range of PF Rx-Queues. 1b = The VSI is assigned a scattered range of PF Rx-Queues.
	28.1-29.7	Reserved	Reserved. Must be zero.
	30-61	Rx-Queue Mapping	If *Mapping Method* = 0 (contiguous): **Bits 30.0-31.2:** The first queue allocated for this VSI in the PF space. **Bits 31.7-31.3:** Reserved **Bytes 33-32:** Number of queues allocated to this VSI. Used to define the number of queues associated with this VSI **Bytes 61-34:** Reserved If *Mapping Method* = 1 (scattered): For each of the queues in the VSI, defines the actual queue in the PF according to the following encoding: For queue 'n' of the VSI offsets 30+2n - 31+2n are used the define the mapping to PF queues. For example for queue 0: • **Bits 30.0-31.2:** The PF queue matching allocated to queue 'n' of the VSI. For non allocated queue, a value of 0x7FF should be set. • **Bits 31.3-31.7:** Reserved. The same mapping is used for the next queues. In this method, up to 16 queues can be assigned. ***Note:*** For VSIs assigned to VFs for which VPLAN_RX_QBASE.*VFQTABLE_ENA* is set, only the scattered method can be used. ***Note:*** The first queue (in both modes) is the default queue of the VSI to which packets not queued by any filter are sent.
	62-77	Number and Offset of Rx-Queue per TCs	Fixes the number of queue pairs assigned to the VSI for each traffic class and the offset of these queues. **Bits 62.0-63.2:** Queue offset for TC0. **Bits 63.3-63.6:** Number of queues allocated to TC0. The actual number is 2^n. The allowed number of queues is: 1, 2, 4, 8, 16, 32, 64, 128. **Bit 63.7:** Reserved. ***Note:*** If no queues needs to be associated to a TC, the queue offset should be set to 0 and the number of queues to 0 (1 queue). This way, traffic associated with this TC is sent to the default queue. The following addresses are used with the same format for the next TCs: **Bytes 64-65:** TC1 **Bytes 66-67:** TC2 **Bytes 68-69:** TC3 **Bytes 70-71:** TC4 **Bytes 72-73:** TC5 **Bytes 74-75:** TC6 **Bytes 76-77:** TC7

Table 7-41. Add VSI Command Buffer [continued]

Category	Byte.Bit	Field	Description
Queuing Options	78.1-78.0	RSS LUT Selection	RSS LUT used. 00b = VSI LUT 01b = Reserved 10b = PF LUT 11b = Global LUT (one of 16 Global LUTs)
	78.5-78.2	RSS Global LUT	If *RSS LUT Selection* = 11b (Global LUT), selects the LUT assigned to this VF.
	78.7-78.6	Hash Scheme	Hash scheme. 00b = Toeplitz Hash 01b = Symmetric Toeplitz 10b = Simple XOR 11b = Reserved
	79.4-79.0	TC Override Option	TC override option.
	79.6-79.5	Reserved	Reserved.
	79.7	Profile Override TC	Profile override TC.
	80.0	Enable PE Filtering	Enable PE filtering.
	80.1-83.7	Reserved	Reserved for future queuing parameters. Must be zero.
Outer UP Mapping	84-87	Egress Inner UP to Outer Translation Table	Defines the UP translation table for transmit packets from inner to outer UP according to the following list: **Bits 84.0-84.2:** Outer UP set if inner UP is 0. **Bits 84.3-84.5:** Outer UP set if inner UP is 1. **Bits 84.6-85.0:** Outer UP set if inner UP is 2. **Bits 85.1-85.3:** Outer UP set if inner UP is 3. **Bits 85.4-85.6:** Outer UP set if inner UP is 4. **Bits 85.7-86.1:** Outer UP set if inner UP is 5. **Bits 86.2-86.4:** Outer UP set if inner UP is 6. **Bits 86.5-86.7:** Outer UP set if inner UP is 7. **Byte 87:** Reserved. This map is used to translate the 802.1P user priority bits on the inner UP to outer UP. To get a fixed outer UP, all the values should be set to the requested outer UP.
ACL	88.0-88.3	Rx Profile Miss Default Action	Selects one of four predefined actions to be applied in case a receive packet does not match any ACL profile.
	88.4-88.7	Rx Table Miss Default Action	Selects one of four predefined actions to be applied in case a receive packet does not match any ACL table entry.
	89.0-89.3	Tx Profile Miss Default Action	Selects one of four predefined actions to be applied in case a transmit packet does not match any ACL profile.
	89.4-89.7	Tx Table Miss Default Action	Selects one of four predefined actions to be applied in case a transmit packet does not match any ACL table entry.

Table 7-41. Add VSI Command Buffer [continued]

Category	Byte.Bit	Field	Description
Flow Director	90.0	Flow Director Enable	Enable Flow Director filtering.
	90.1	Tx Auto-Evict Enable	Enables eviction of ATR filters if a FIN/RST packet is sent.
	90.2	Reserved	Reserved.
	90.3	Flow Director Programming Enable	Enable Flow Director programming.
	90.4-91.7	Reserved	Reserved.
	92-93	Max Dedicated	Max Number of allocated Flow Director filters to a VSI.
	94-95	Max Shared	Max number of shared Flow Director filters any VSI can program. ***Note:*** If both dedicated and shared values are zero, the VSI cannot program filters.
	96.0-97.2	Default Queue	This field defines the receive queue index within the VSI of the default FD. Permitted values are in the range of the queues of the VSI.
	97.3	Reserved	Reserved.
	97.6-97.4	Default Queue Group	The *ToQueue* parameter associate a target queue or a queue group to the FD filter: 0 = The default filter assign a target queue defined by the QINDX 1...7 = The default filter assign a region of queues. The region base equals to QINDX and the region size equals to 2^(ToQueue).
	97.7	Reserved	Reserved.
	98.0-99.2	Reporting Queue	This field defines the receive queue index of the programming VSI on which the FD filter completion status is reported when the *COMP_Queue* flag in the programming descriptor is set.
	99.3	Reserved	Reserved.
	99.6-99.4	Default Priority	The priority of the default QINDX action.
	99.7	Default Drop	When set, a packet that miss the FD filters is dropped.
PASID Section	102.3-100.0	PASID ID	PASID for this VSI.
	103.6-102.4	Reserved	Reserved.
	103.7	PASID ID Valid	PASID ID is valid.
Reserved	104-127	Reserved	Reserved.

Table 7-42 describes the Add VSI response.

Table 7-42. Add VSI Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0210	Command opcode.
Datalen	4-5	0x80	Length of buffer.
Return Value/VFID	6-7		Return value. The following error values can be returned: ENOSPC = If there are not enough resources to assign a VSI (in case of VSI number invalid) or part of the requested resources. EINVAL = The VSI is already enabled ENOENT = Index is invalid for one of the VSI resource parameters. EACCES = PF does not own one of the VSI resource parameters.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
VSI	16-17		Returns the assigned VSI number. If VSI is provided in command, the same value is returned. **Bit 17.7:** VSI is valid (always valid) **Bits 17.6-17.2:** Reserved **Bits 17.1-16.7:** VSI Number
Extended Status	18-19		If this value is not zero and there is no error code, the command succeeded but with warnings.
VSIs Allocated from Guaranteed	20-21		Number of VSIs that were either allocated using the Allocate Resource command, or allocated and enabled using the Add VSI command, both from the VSIs pre-allocated in NVM.
Total VSIs Un-Allocated	22-23		Total number of VSIs still unallocated and not reserved by any function.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

7.8.12.3.2 Update VSI (0x0211)

This command is used to update the parameters of an existing VSI. A function can update only a VSI it controls. Only sections defined as valid in Bytes 0-1 of the buffer are updated.

Table 7-43. Update VSI Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0211	Command opcode.
Datalen	4-5	0x80	Length of buffer.
Return Value/VFID	6-7		Return value. Zeroed by device driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
VSI Number	16-17		**Bit 17.7:** VSI is valid (should always be set) **Bits 17.6-17.2:** Reserved **Bits 17.1-16.7:** VSI Number
Reserved	18-23		Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

The command buffer and the completion buffer used to update a VSI is the same as the command buffer of the Add VSI command (Table 7-41). Specific limitations are listed in the table.

All the filters set before the VSI type updates are kept, and the software device driver should guarantee they are compliant with the new VSI parameters.

Table 7-44. Update VSI Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0210	Command opcode.
Datalen	4-5	0x80	Length of buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. The following error values can be returned: EACCES = If the VSI is not owned by this PF or PF does not own one of the VSI resource parameters. EINVAL = The VSI was not enabled. ENOSPC = There are not enough resources to allocate new resources requested. ENOENT = Index is invalid for one of the VSI resource parameters.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
VSI Number	16-17	0x0	Copied from command.
Reserved	18-19		Reserved.
VSIs Allocated from Guaranteed	20-21		Number of VSIs that were either allocated using the Allocate Resource command, or allocated and enabled using the Add VSI command, both from the VSIs pre-allocated in NVM.
Total VSIs Un-Allocated	22-23		Total number of VSIs still unallocated and not reserved by any function.

Table 7-44. Update VSI Response [continued]

Name	Byte.Bit	Value	Remarks
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

7.8.12.3.3 Get VSI Parameters (0x0212)

This command is used to get the parameters of an existing VSI. A function can query only a VSI it controls.

Table 7-45. Get VSI Parameters Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0212	Command opcode.
Datalen	4-5	0x80	Length of buffer.
Return Value/VFID	6-7		Return value. Zeroed by device driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
VSI Number	16-17		**Bit 17.7:** VSI is valid (should always be set) **Bits 17.6-17.2:** Reserved **Bits 17.1-16.7:** VSI Number
Reserved	18-23		Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

Table 7-46. Get VSI Parameters Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0212	Command opcode.
Datalen	4-5	0x80	Length of buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. The following error values can be returned: ENOENT = If the SEID do not point to a valid VSI. EACCES = If the VSI is not owned by this PF.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
VSI Number	16-17		VSI Number (copied from command).
VF Function Number	18		Defines the VF function to which this VSI connects. Valid only if Function Type is VF. Should be ignored if VSI type is not VF. ***Note:*** The VF number here is the absolute VF number (0-255) and not the number relative to the PF first VF.

Table 7-46. Get VSI Parameters Response [continued]

Name	Byte.Bit	Value	Remarks
Status Flags	19		**Bits 1:0:** VSI type 00b = VF 01b = VMDq2 (a.k.a. VM) 10b = PF 11b = EMP/MNG **Bit 7:2:** Reserved.
VSIs Allocated from Guaranteed	20-21		Number of VSIs that were either allocated using the Allocate Resource command, or allocated and enabled using the Add VSI command, both from the VSIs pre-allocated in NVM.
Total VSIs Un-Allocated	22-23		Total number of VSIs still unallocated and not reserved by any function.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

Table 7-47 describes the response buffer received when querying a VSI.

Table 7-47. Get VSI Parameters Response Buffer

Byte.Bit	Description
0-95	Same parameters as described in Table 7-41.
96-127	Reserved.

7.8.12.3.4 Free VSI (0x0213)

This command is used to remove a VSI. A function can remove only a VSI it owns. The driver should make sure that every queue in a VSI is disabled before a VSI is removed and that no rules pointing to this VSI exists.

As part of this command, the firmware should clear all the Flow Director filters assigned to this VSI by applying the flow of the Clear FD Table AQ command (0x0B06) described in Section 7.10.12.1 and clear the RSS key of the VSI as if the Set RSS Key (0x0B02) command was received with an all-zero key.

Table 7-48. Free VSI Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0213	Command opcode.
Datalen	4-5	0x0	Reserved.
Return Value/VFID	6-7		Return value. Zeroed by device driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
VSI	16-17	VSI	The VSI to remove **Bit 17.7:** VSI is valid (should always be set) **Bits 17.6-17.2:** Reserved **Bits 17.1-16.7:** VSI Number
Command Flags	18-19		**Bits 19.7-18.1:** Reserved **Bit 18.0:** Keep VSI Allocation If set, the VSI stays part of the PF allocated resources. Otherwise, it is returned to the shared pool.
Reserved	20-31		Reserved. Must be zero.

Table 7-49. Free VSI Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0213	Command opcode.
Datalen	4-5	0x0	Reserved.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. The following error values can be returned: EACCES = If the element is not owned by this PF. ENOENT = VSI number is invalid.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-19		Reserved.
VSIs Allocated from Guaranteed	20-21		Number of VSIs that were either allocated using the Allocate Resource command, or allocated and enabled using the Add VSI, both from the VSIs pre-allocated in NVM.
Total VSIs Un-Allocated	22-23		Total number of VSIs still unallocated and not reserved by any function.
Reserved	24-31		Reserved.

7.8.12.4 Storm Control Commands (0x028x)

7.8.12.4.1 Set Storm Control Configuration (0x0280)

This command is used to configure the Storm Control Mechanism of the port.

This command can be applied only by the function that controls the physical port.

Table 7-50. Set Storm Control Configuration Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0280	Command opcode.
Datalen	4-5	0x0	Length of buffer.
Return Value/VFID	6-7		Return value. Zeroed by device driver.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Broadcast Threshold	16-19	0x0	**Bits 18:0:** Traffic Upper Threshold Size Represents the upper threshold for broadcast storm control. **Bits 31:19:** Reserved
Multicast Threshold	20-23	0x0	**Bits 18:0:** Traffic Upper Threshold Size Represents the upper threshold for multicast storm control. **Bits 31:19:** Reserved
Storm Control Control	24-27	0x0	**Bit 0:** MDIPW Drop multicast packets (excluding flow control and manageability packets) if multicast threshold is exceeded in previous window. **Bit 1:** MDICW Drop multicast packets (excluding flow control and manageability packets) if multicast threshold is exceeded in current window. **Bit 2:** BDIPW Drop broadcast packets (excluding flow control and manageability packets) if broadcast threshold is exceeded in previous window. **Bit 3:** BDICW Drop broadcast packets (excluding flow control and manageability packets) if broadcast threshold is exceeded in current window. **Bits 7:4:** Reserved **Bits 27:8:** INTERVAL - BSC/MSC Time-interval-specification The interval size for applying Ingress Broadcast or Multicast Storm Control. Interrupt decisions are made at the end of each interval (and most flags are also set at interval end). **Bits 31:28:** Reserved
Reserved	28-31		Reserved.

Table 7-51. Set Storm Control Configuration Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0280	Command opcode.
Datalen	4-5	0x0	Length of buffer.
Return Value/VFID	6-7		Return value. There is no specific error code for this command.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-31		Reserved.

7.8.12.4.2 Get Storm Control Configuration (0x0281)

This command is used to query the configuration of the storm control mechanism. Note that the current status of the storm control is received via the SCSTS register.

Table 7-52. Get Storm Control Configuration Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0281	Command opcode.
Datalen	4-5	0x0	Reserved
Return Value/VFID	6-7	0x0	Must be zero.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-31		Reserved.

Table 7-53. Get Storm Control Configuration Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0281	Command opcode.
Datalen	4-5	0x0	Length of buffer.
Return Value/VFID	6-7		Return value. There is no specific error code for this command.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Broadcast Threshold	16-19	0x0	**Bits 18:0:** Traffic Upper Threshold Size Represents the upper threshold for broadcast storm control. **Bits 31:19:** Reserved
Multicast Threshold	20-23	0x0	**Bits 18:0:** Traffic Upper Threshold Size Represents the upper threshold for multicast storm control. **Bits 31:19:** Reserved

Table 7-53. Get Storm Control Configuration Response [continued]

Name	Byte.Bit	Value	Remarks
Storm Control Control	24-27	0x0	**Bit 0:** MDIPW Drop multicast packets **Bit 1:** MDICW Drop multicast packets **Bit 2:** BDIPW Drop broadcast packets **Bit 3:** BDICW Drop broadcast packets **Bit 4:** BIDU BSC Includes Destination Unresolved packets: **Bits 7:5:** Reserved **Bits 27:8:** Interval - BSC/MSC Time-interval-specification The interval size for applying Ingress Broadcast or Multicast Storm Control. **Bits 31:28:** Reserved
Reserved	28-31		Reserved.

7.8.12.5 Switch Recipes Configuration (0x029x)

7.8.12.5.1 Add Recipe (x0x290)

This command adds a recipe to the 64-entry recipe table. The recipe can include multiple entries to create a composite rule. After the recipe is added, it should be associated with packet profiles using the Set Recipes to Profile Association command.

If the recipe exists, it is updated. It is the responsibility of software not to change recipes pointing to populated rules, unless the change is to add a result to the recipe to be able to use it as a sub-recipe.

Table 7-54. Add Recipe Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0290	Command opcode.
Datalen	4-5	Buffer Length	(Number of recipe +1) * 32
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of Sub-Recipes	16-17		Number of sub-recipe entries in buffer.
Reserved	18-23	0x0	Reserved. Must be zero.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

The buffer in the command is as follows:

Table 7-55. Add Recipe Buffer[1]

Byte	Description
0 + n *64	Recipe index.
1 + n *64 - 3 + n *64	For command: Reserved. For response: 0: Recipe was updated. 23:1: Reserved.
4 + n *64 - 11+ n *64	Bitmap of the recipe indexes associated with this recipe A recipe can make use of already programmed entries that might not be part of the recipes described in this command.
12+ n *64 - 15+ n *64	Reserved.
16 + n *64 - 43 + n *64	Content of recipe as described in Table 7-9.
44 + n *64 - 63 + n *64	Reserved.

1. Repeated n times, based on the number of programmed recipe entries.

Table 7-56. Add Recipe Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0290	Command opcode.
Datalen	4-5	Buffer Size	
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. The following error values can be returned: EACCES = If one of the resources is not owned by this PF. ENOENT = Bad resource index.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of Sub-Recipes	16-17		Number of sub-recipe entries in buffer.
Reserved	18-23		Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

7.8.12.5.2 Set Recipes-to-Profile Association (0x0291)

This command is use to associate recipes with packet profiles.

Table 7-57. Set Recipes-to-Profile Association Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0291	Command opcode.
Datalen	4-5	Buffer Length	Reserved.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Profile ID	16-17		The profile to associate recipes to.
Reserved	18-23	0x0	Reserved. Must be zero.
Recipe Association	24-31		Bitmap of the recipe indexes associated with this profile. The bitmap should include both root and non-root recipe associated with the profile.

Table 7-58. Set Recipes-to-Profile Association Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0291	Command opcode.
Datalen	4-5	0x0	Reserved.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. The following error values can be returned: EACCES = If one of the resources is not owned by this PF. ENOENT = Bad resource index. ERANGE = A root recipe is added to the association list without its dependent recipes.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-31		Reserved.

7.8.12.5.3 Get Recipe (0x0292)

This command returns a recipe and all the associated entries.

Table 7-59. Get Recipe Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0292	Command opcode.
Datalen	4-5	Buffer Length	Should be 4 KB.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of Rules	16-17		Reserved in command. Filled in response.
Recipe Root Number	18-19		Index of the root recipe to return (0-63). All the child recipes of this root recipe are also returned. A value of 0xFFFF means to start from the very first root and return all the active recipes with their children. ***Note:*** The recipes returned are either allocated to the PF or persistent.
Reserved	20-23	0x0	Reserved. Must be zero.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

Table 7-60. Get Recipe Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0292	Command opcode.
Datalen	4-5	0x0	Reserved.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. The following error values can be returned: ENOENT = Bad resource index.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of Rules	16-17		Reserved in command. Filled in response according to the number of rules associated with the requested root ID.
Recipe Number	18-19		Index of root recipe requested.
Reserved	20-23		Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

The buffer in the response is as follows:

Table 7-61. Get Recipe Response Buffer[1]

Byte.Bit	Description
0 + n *64	Recipe Index (the recipes are ordered numerically). The Root recipe is the last one in the list
1 + n *64 - 3 + n *64	Reserved.
4 + n *64 - 11+ n *64	Bitmap of the recipe indexes associated with this recipe (all the recipe that participates in the creation of the requested recipe). These recipe should appear earlier in the buffer.
12+ n *64 - 15+ n *64	Reserved.
16 + n *64 - 43 + n *64	Content of recipe as described in Table 7-9.
44 + n *64 - 63 + n *64	Reserved.

1. The list is repeated *Number of Rules* times.

7.8.12.5.4 Get Recipes-to-Profile Association (0x0293)

This command is use to read the recipes associated with a packet profile.

Table 7-62. Get Recipes-to-Profile Association Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0293	Command opcode.
Datalen	4-5	Buffer Length	Reserved.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Profile ID	16-17		The profile to read the association of.
Reserved	18-31	0x0	Reserved. Must be zero.

Table 7-63. Get Recipes-to-Profile Association Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0293	Command opcode.
Datalen	4-5	0x0	Reserved.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. The following error values can be returned: EACCES = if one of the resources is not owned by this PF. ENOENT = Bad resource index.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Profile ID	16-17		The profile to read the association of.
Reserved	18-23	0x0	Reserved.
Recipe Association	24-31		Bitmap of the recipe indexes associated with this profile.

7.8.12.6 Switch Rules Population Commands (0x02Ax)

7.8.12.6.1 Add Switch Rules (0x02A0)

This command is used to populate the switch. It can be used to program a combination of lookup rules + actions, large actions or VSI lists.

A single buffer can include multiple commands from different types. The commands are applied in order, so if a rule points to a large action or a VSI list, the list and the large action should be programmed before the lookup + action.

If a rule already exists and needs to be updated, the Update Switch Rules command should be used.

Table 7-64. Add Switch Rules Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x02A0	Command opcode.
Datalen	4-5	Buffer Length	Reserved.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of Rules	16-17		Number of rules in buffer.
Reserved	18-23	0x0	Reserved. Must be zero.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

Table 7-65. Add Switch Rules Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x02A0	Command opcode.
Datalen	4-5	0x0	Reserved.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. The following error values can be returned: EACCES = If one of the resources is not owned by this PF. ENOSPC = If one of the filters cannot be allocated. EINVAL = Invalid parameters or rule already exists. ENOENT = Bad resource index. ***Note:*** If an ENOSPC error is returned, check the return buffer for the exact list of filters not allocated. Other errors indicates that the entire command was not executed.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-23		Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

The content of the return buffer is the same as the input buffer. The *Status* field and the LUT index are updated as part of the response. The first filter allocation that fails due to lack of resources stops the command.

7.8.12.6.1.1 Add Switch Rules Buffer Format

The buffer contains a “Number of Rules” structures with the following format:

```
struct
{
#define T_LOOKUP_RX         0x0 // Used to add a rule for traffic from network.
#define T_LOOKUP_TX         0x1 // Used to add a rule for traffic from VSI.
#define T_LARGE_ACTION      0x2 // Used to define content of large action.
#define T_VSI_LIST_SET      0x3 // Used to set VSIs in a multicast VSI list.
#define T_VSI_LIST_CLEAR    0x4 // Used to clear VSIs from a multicast VSI list or to
                                // clear the entire list.
#define T_PRUNE_LIST_SET    0x5 // Used to set VSIs in a prune VSI list.
#define T_PRUNE_LIST_CLEAR 0x6 // Used to clear VSIs from a prune VSI listor to clear
                                // the entire list.

    __le16 Type;     // One of #defines above.
    __le16 Status;   // Return status. For Add/Update indicates the resource was
                     // successufully allocated/updated. For Get, always succeed for
                     // entries in range. A status of zero means a success. A non
                     // zero status reflects one of the errors listed in Table 7-65
                     // for return value.

    union
    {
        struct
        {
            __le16 RID;        // Recipe ID.
            __le16 Source;     // Source VSI for LOOKUP_TX and source port for
                               // LOOKUP_RX. This field is used by the hardware to
                               // derive the switch ID (based on source of packet).
                               // Reserved for Get command.
            __le32 Action;     // Action as defined in Table 7-12.
            __le16 Index;      // LUT index. Return value for Add, parameter for
                               // Get/Update.
            __le16 HeaderLen; // Header length (in bytes) for add command. Should be
                               // zero for remove and update command. Should be 24 for
                               //  query. Should be aligned to four bytes.
            u8 HeaderData[0]; // Header data, array of HeaderLen for add and update
                               // commands. For query, contains the content of the
                               // FLU. Should be null for remove command.
        } LOOKUP; // LOOKUP_TX and LOOKUP_RX

        struct
        {
            __le16 Index;      // Index in Large action table.
            __le16 Size;       // 0/1/2/4 - 0 is used for free commands.
            __le32 Action[0]; // Array of Size of actions as defined in Table 7-14.
        } LARGE_ACTION;

        struct
        {
            __le16 Index;         // Index of VSI/Prune list.
            __le16 NumberOfVSIs; // Should be zero to clear the entire list.
```

```
            __le16 VSI[0];       // Array of NumberOfVSIs VSI numbers.
        } VSI_LIST; // Used for VSI_LIST_SET, VSI_LIST_CLEAR, PRUNE_LIST_SET and
                    // PRUNE_LIST_CLEAR.

        struct
        {
            __le16 Index;       // Index of VSI/Prune list.
            __le8 VSI_lIST[96]  // Bitmap of VSI list.
        } VSI_LIST_QUERY; //Used for queries of VSI_LIST_SET/CLEAR

    u8 Padding[0]; //Array of padding bytes
    } Data;
} PopulationEntry;
```

Note: The option to clear an entire VSI list by using an empty VSI_LIST and a PRUNE_LIST_CLEAR action is available only as part of a Remove Switch Rules command.

The buffer header should be a valid packet header.

7.8.12.6.1.2 Population Example

The following is an example of a population request to implement a rule that upon reception of a multicast packet with DA = DA1, it forwards it to VSI1 and VSI3 and mirrors it to VSI12. It is assumed that all the resources are already allocated.

The buffer consists of 3 rules:

- Populate a VSI list with VSI1 and VSI3.
- Create a large action with two actions:
  - Forward to VSI list.
  - Mirror to VSI.
- Create a lookup rule with the packet header and with an action pointing to the larger action mentioned in the previous step.

Assume that VSI List VLn and Large Action LAk are the resources the software device driver chooses to use for this rule and assume RID = 0 is the MAC forwarding recipe used.

To create this, the following buffer is created:

```
struct
{
    __le16 Type = T_VSI_LIST_SET;
    __le16 Status; // filled in response
        struct
        {
            __le16 Index = n;          //VLn
            __le16 NumberOfVSIs = 2;
            __le16 VSI[0] = {1,3};     // VSI1 and VSI3
        } VSI_LIST;
} Entry_1;

struct
{
    __le16 Type = T_LARGE_ACTION;
    __le16 Status; // filled in response
    struct
```

```
        struct
        {
            __le16 Index = k; // LAk
            __le16 Size = 2;  // 2 actions in large action
            __le32 Action[0] = {VSI List forward action(000), Mirror action(011),...}
        } LARGE_ACTION;
    } Data;
} Entry_2;

struct
{
    __le16 Type = T_LOOKUP_TX;
    __le16 Status; // filled in response
    struct
        struct
        {
            __le16 RID = 0 ; //MAC Forwarding
            __le16 Source = 1; // VSI1 is one of the VSIs in the VEB
            __le32 Action = {Pointer Action};
            __le16 Index; // filled in response
            __le16 HeaderLen = 14; //Header length of an L2 packet
        u8 HeaderData[0] = {DA1,Dummy SA, any EtherType }; // Do not use an EtherType
                                                          // of IP if IP header not
                                                          // provided.
        } LOOKUP;
   } Data;
} Entry_3;
```

7.8.12.6.2 Update Switch Rules (0x02A1)

Table 7-66. Update Switch Rules Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x02A1	Command opcode.
Datalen	4-5		Length 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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of Filters	16-17		Number of filters in buffer.
Reserved	18-23		Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

Data buffer format is described in Section 7.8.12.6.1.1. For update commands, the lookup rules should be accessed by index.

Note: For Large Action and VSI list updates, the Add Switch Rules command and this command can be used interchangeably.

Table 7-67. Update Switch Rules Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x02A1	Command opcode.
Datalen	4-5	0x0	Reserved.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. The following error values can be returned: EACCES = If one of the resources is not owned by this PF. ENOENT = If the lookup rule does not exists.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-23		Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

7.8.12.6.3 Remove Switch Rules (0x02A2)

Table 7-68. Remove Switch Rules Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x02A2	Command opcode.
Datalen	4-5		Length 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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of Filters	16-17		Number of filters in buffer.
Reserved	18-23		Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

Data buffer format is described in Section 7.8.12.6.1.1. For remove commands, the lookup rules should be accessed by index.

Table 7-69. Remove Switch Rules Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x02A2	Command opcode.
Datalen	4-5	0x0	Reserved
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. The following error values can be returned: EACCES = If one of the resources is not owned by this PF. ENOENT = If the lookup rule does not exists.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-23		Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

7.8.12.6.4 Get Switch Rules (0x02A3)

Table 7-70. Get Switch Rules Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x02A3	Command opcode.
Datalen	4-5		Length 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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of Filters	16-17		Number of filters in buffer.
Reserved	18-23		Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

Data buffer format is described in Section 7.8.12.6.1.1. For Get command, the lookup rules should be accessed by index. In the case of Get command there is no difference between T_VSI_LIST_CLEAR and T_VSI_LIST_SET types, and no difference between T_PRUNE_LIST_CLEAR and T_PRUNE_LIST_SET types. For large actions, the buffer size should be set according to the expected size of the action.

Table 7-71. Get Switch Rules Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x02A3	Command opcode.
Datalen	4-5		Length of command buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. The following error values can be returned: EACCES = If one of the resources is not owned by this PF. ENOENT = If the lookup rule does not exists.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-23		Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

7.8.12.6.5 Clear PF Configuration (0x02A4)

This command clears all the configuration of the PF, including all VSIs and all Flow Director filters allocated to this PF.

The command does not release resources allocated to the function and the switch ID allocation of the port.

The following resources are removed (but not deallocated) as part of the command:

- Port Configuration
- Switch Rules
- Mirror Rules
- VSI lists
- VSIs
- VEB Counters
- FD Filters

Resource management, Recipes (and Recipe to Profile associations), PE filters are not cleared by this command.

The original topology (VSI + port MAC filter) is not initiated after this command and should be built by the software device driver.

Table 7-72. Clear PF Configuration Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x02A4	Command opcode.
Datalen	4-5	0x0	Reserved.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-31		Reserved.

Table 7-73. Clear PF Configuration Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x02A4	Command opcode.
Datalen	4-5		Length of return buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-31		Reserved.

7.8.12.7 Mirroring Commands (Opcode 0x026x)

The mirroring behavior is described in Section 7.8.5.4.1.

7.8.12.7.1 Add/Update Mirror Rule (0x0260)

This command is used to add a port mirror rule. Rules can be added for any ingress or egress port, virtual (VSI) or physical (Ethernet port). Lookup based mirroring (a.k.a. VLAN mirroring), is programmed via the regular Add Switch Rules command (Section 7.8.12.6.1).

Table 7-74. Add/Update Mirror Rule Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0260	Command opcode.
Datalen	4-5		Length of buffer. Should be equal to 2 * *Number of Mirrored Entries*.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Rule ID	16-17		**Byte 17:** Reserved **Bit 16.7:** Rule ID Valid **Bit 16.6:** Reserved **Bits 16.5-16.0:** Rule ID If *Rule ID Valid* is set, the requested *Rule ID* is used. If the rule is already in use, the new mirroring rules are added to this rule. The rule should be owned by the PF requesting the it. If *Rule ID Valid* is cleared, a new *Rule ID* is assigned from the shared pool of rules.
Rule Type	18-19		**Bits 2:0:** Rule Type 000b = Reserved. 001b = Virtual port ingress mirroring. All traffic sent from the VSIs in the buffer 010b = Virtual port egress mirroring. All traffic received by the VSIs in the buffer 011b = Reserved (VLAN mirroring in previous products). 100b = Reserved (All ingress traffic in previous products). 101b = Reserved (All egress traffic n previous products). 110b = Physical Port ingress mirroring. All traffic received by this LAN port. 111b = Physical Port egress mirroring. All traffic sent by this LAN port. **Bits 15:3**: Reserved. ***Note:*** If the *Rule Type* is 110b or 111b (ingress/egress port mirroring), there is no associated buffer. The flags in bytes 0-1 should be set accordingly and *Number of Mirrored Entries* should be zero.
Number of Mirrored Entries	20-21		Defines the number of VSIs that should be mirrored. The values are in the command buffer.
Destination VSI	22-23		Defines the VSI to which the packets matching the mirror rule will be mirrored. **Bits 15:10:** Reserved **Bits 9:0:** VSI Number
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

The Command Buffer is built as *Number of Mirrored Entries* entries of two bytes each containing a single VSI/VLAN, depending on the rule requested as described in Table 7-75.

Table 7-75. Add/Update Mirror Rule Command Buffer

Byte.Bit	Description
1.7	Action 0b = Remove VSI from mirror rule. 1b = Add VSI to mirror rule.
1.6-1.2	Reserved.
1.1-1.0	Mirrored VSI.

Table 7-76 describes the Add/Update Mirror rule response (with no buffer).

Table 7-76. Add/Update Mirror Rule Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0260	Command opcode.
Datalen	4-5	0x0	No return buffer.
Return Value/VFID	6-7		Return value. The following error values can be returned: ENOENT = If the mirrored or mirroring VSI do not point to a valid VSI. ENOSPC = If there are not enough resources to assign an mirror rule. EACCES = If the *Rule ID* is not owned by this PF. EEXIST = VSI is already mirrored by another rule. ***Note:*** If an error is returned, the entire command was not executed.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Rule ID	16-17		**Byte 17:** Reserved **Bit 16.7:** Rule ID valid (always set) **Bit 16.6:** Reserved **Bits 16.5-16.0:** Rule ID Defines the *Rule ID* that is returned in the receive descriptor. This number is assigned by the firmware and should be used as a handle when requesting deletion of an existing rule. If the *Rule ID Valid* flag was set in the command, the returned *Rule ID* is the same as the requested one.
Reserved	18-23		Reserved.
Data Address High	24-27	Buffer Address	Address of Buffer
Data Address Low	28-31	Buffer Address	

7.8.12.7.2 Delete Mirror Rule (0x0261)

This command is used to delete an existing mirror rule.

Table 7-77. Delete Mirror Rule Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0261	Command opcode.
Datalen	4-5	0x0	
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Rule ID	16-17		**Byte 17:** Reserved **Bit 16.7:** Rule ID valid (always set) **Bit 16.6:** Reserved **Bits 16.5-16.0:** Rule ID Defines the Rule ID to delete.
Reserved	18-19		Reserved.
Command Flags	20-21		**Bits 21.7-20.1:** Reserved **Bit 20.0:** Keep m Allocation If set, the VSI stays part of the PF allocated resources. Otherwise, it is returned to the shared pool.
Reserved	22-31	Buffer Address	Reserved.

Table 7-78 describes the Delete Mirror rule response (with no buffer)

Note: When a mirror rule is shared (VSIs of multiple PFs are mirrored through the same rule), this command should not be used. Instead, each PF should remove its VSIs using the Update Mirror rule command and this command should be used only by the last PF to remove the rule itself.

Table 7-78. Delete Mirror Rule Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0261	Command opcode.
Datalen	4-5	0x0	
Return Value/VFID	6-7		Return value. The following error values can be returned: EINVAL = If the Rule ID does not exist. ENOENT = If the Rule ID is not owned by this PF.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-31	Buffer Address	Reserved.

7.8.12.8 Default Configuration at Init Time

On each function, a single VSI is initiated with a distinct switch ID. This VSI is set in VLAN promiscuous mode with loopback disabled and source pruning enable.

On each VSI, a single filter is added to allow reception of the default MAC Address of the port.

7.9 ACL (Ternary Classifier)

7.9.1 Overview

7.9.1.1 General

The Packet Processor pipeline in the E810 provides a programmable Ternary Classifier stage for implementing functions such as ACL (Access Control List), IP LPM (Longest Prefix Match), and so on. The Ternary Classifier is SDN/NFV-enabled, supporting a wide range of network protocols, both standard-based and proprietary. The architecture is fully programmable, protocol-agnostic and action-agnostic, capable of adapting to future protocol headers and use cases.

Since the architecture of the Ternary Classifier is protocol-agnostic and action-agnostic, it can be programmed to serve as a switch extension and thus provide packet forwarding based on programmable rules, augmenting the capabilities of the switch.

The ACL's block logical location in the E810 data-path is illustrated in Figure 7-19.

On the receive side, the ACL is located on an internal egress port of the switch. This means that it can base its decisions on the switch switching decision (destination information). The ACL block gets the packet's header as received by the switch. Any potential transposition (for example, VLAN stripping) occurs in the data path following the packet processing by the ACL block.

On the transmit side, the ACL is location on an internal ingress port of the switch. This means that the ACL block can base its decisions based on the source information. The ACL block gets packet header in its final transmitted form. Any potential transposition (for example, VLAN insertion) occurs in the data path prior to the packet processing by the ACL block.

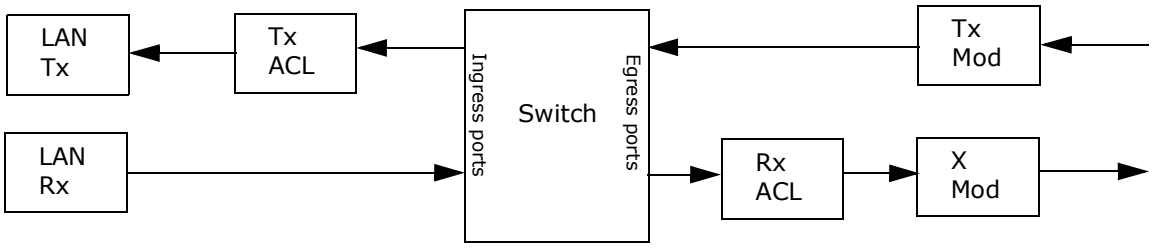


Figure 7-19. ACL Logical Location in the Data Path

7.9.1.2 Features

- Provides ACL lookup tables:
  - Table contain association between ACL entries and actions.
  - ACL match is done using up to 16 TCAM blocks.
  - Each TCAM block has 512 entries of 40 bits.
  - TCAM cascading.
- Hit result of consecutive range of TCAM block indexes can be used to create an ACL table with wider rules (multiple of 40 bits).
  - TCAM stacking.
- Multiple TCAM blocks, with consecutive index range, can be stacked to create a table with more than 512 rules (cascaded table can be stacked as well).
- 128 lookup profiles:
  - Standard FlexiPipe method of profile selection.
  - One profile is used for by pass (profile number 1).
  - Eight range checkers per profile:
    - Range checker works on a word (16-bit value).
    - Range checker apply a mask on the value and checks if it is between a start value and an end value.
  - Profile controls the extractions:
    - Extraction of a field vector (FV) built from 32 words.
    - Build of three arrays from the FV:
      - 32-byte array with also contain the Profile ID and the eight range checks result (pass/fail per each checker).
      - 32-word array.
      - 16 double word array.

      Profiles selects a scenario.
- 64 Scenario
  - Controls generation of key per TCAM block.
  - Controls mask of entries per TCAM block in 64 entries granularity (bit per 64 entries).
  - Controls TCAM block arrangement.
  - Controls action memories to TCAM block association.
- Action memories:
  - 20 action memories with 512 pair of actions each.
  - Multiple action memories CAN be associated with single TCAM block.
  - Key action types.
    - Generic MDID 0-4

    - flowID_word_0/1
    - RX_descr_structure_prof_idx
    - pkt_drop
    - Target/destination VSI
    - Destination Q
    - Count packet (MDID 56)
    - Count bytes (MDID 57)
    - Count packet and bytes (MDID 58)
    - NOP (MDID 55)
  - Counter banks:
  - Four counter banks of 512 40 bit wide counters.
  - Lookup results of a scenario can update one counter in each bank.
- Programming model
  - CSRs
  - Indirect programming access to allow multiple atomic updates.
    - TCAM entry update.
    - Action memory update.
    - Scenario configuration per TCAM block update.
    - Scenario configuration per action memory update.
    - General profile concurrent configuration update.
    - Range checkers profile concurrent configuration update.
  - Dynamic updates.
    - New profiles and scenarios can be added/removed/updated.
    - Actions can be updated/added/removed.

7.9.1.3 Basic Operation and Terms

A ternary classifier is a unique filtering mechanism able to compare a given dataset (referred hereafter as a "key") to a set (referred hereafter as a "table") of multiple entries (referred hereafter as "table entries") at the same time being able to ignore bits in the compare process. This means that a compare operation for each bit in the table entry has three match conditions:

- Match if '0'
- Match if '1'
- Match always (Don't care)

A match in a table entry is achieved only if there is a match on all of the bits of that table entry.

In case of a multiple hit to different table entries, a priority mechanism is used to determine the highest priority-matched entry. The priority mechanism implemented in the E810's ACL block, which selects the lowest hit entry's index in the table. For example, if entries 3, 7, and 11 are hit, the result is entry 3.

Employing the priority mechanism essentially means that a specific key search in a specific entry can yield multiple hits, but only a single result (referred hereafter as a "result entry").

The ternary match and priority mechanisms are implemented using a TCAM block. Multiple TCAM blocks in parallel are used to extend the functionality of a single TCAM block by:

- Extending the number of entries.
- Producing wider entries.
- Doing a match on multiple tables in parallel.

7.9.1.4 Table Configuration Options

7.9.1.4.1 General

The ACL block supports a flexible arrangement of its TCAM blocks to support tables with various key widths and entry amounts. This section depicts the various tools implemented in the ACL block that can be used to manifest tables in different sizes.

The TCAM blocks arrangement is a scenario based configuration (see Section 7.9.1.8), which means that the ACL block can support up to 64 independent TCAM block configurations.

7.9.1.4.2 Table Sizing Support

Since the TCAM blocks used in the design are 40 bits wide, supporting a table with a wider key requires cascading several TCAM blocks (see Section 7.9.1.4.2.1). Each TCAM block cascaded to the other increases the supported key width of the table by 40 bits.

Similarly, since the TCAM blocks used in the design support a total of 512 entries, supporting a table with a larger amount of entries requires stacking several TCAM blocks.

As the E810 implementation uses 16 TCAM blocks, and given the description above, the maximal supported key width for an ACL table in the E810 is 640 bits (16 cascaded TCAM blocks yielding 512 entries), and the maximal supported entry amount for an ACL table in the E810 is 8,192 (16 stacked TCAM blocks and using 40-bit entries).

7.9.1.4.2.1 TCAM Cascade

A TCAM cascade is used to support tables with a key width wider than 40 bits. The TCAM cascade mechanism operates by running several TCAM blocks lookup in parallel and returning a hit in a given entry only if the same entry was hit in all the cascaded TCAM blocks (that is, perform a logic AND operation).

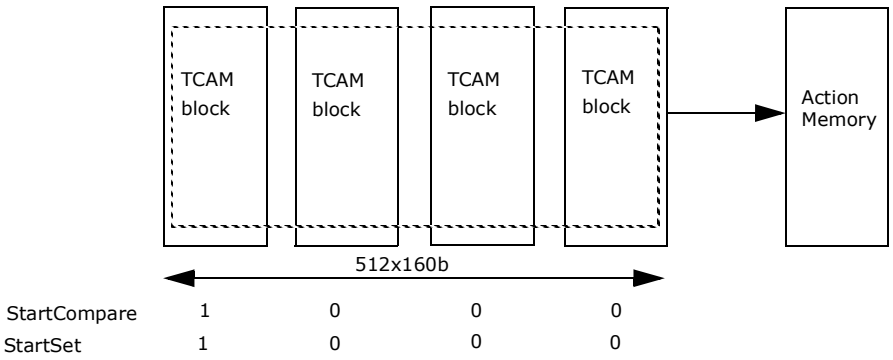


Figure 7-20. TCAM Cascading Example

The ACL block requires cascaded TCAM blocks to be adjacent (i.e., to have consecutive indexes). For example, a table supporting an 80-bit wide key can be implemented using TCAM blocks index 1 and 2, but not using TCAM block 1 and 3.

When cascaded, the hit vector, representing the hit indications for an entire TCAM, is propagated to the next TCAM. The result of that next TCAM block is the logical AND of the previous TCAM block's hit vector and its own local hit vector. That result can also be used in the next TCAM block and so forth to support wider keys.

The basic mechanism that controls TCAM cascading is that each TCAM block can either use its local hit vector or the result of a logical AND between the previous TCAM hit vector and its local hit vector. This setting is controlled by the StartCompare configuration, which is part of the scenario configuration per TCAM (see Section 7.9.2.5.1). When setting StartCompare for a given TCAM block, it uses its local hit vector. When clearing StartCompare for a given TCAM block, it uses the result of a logical AND between the previous TCAM hit vector and its local hit vector.

Given this explanation, the StartCompare setting is set for independent TCAM blocks and for the first TCAM block in a TCAM cascade.

The unified result entry is reflected in the output of the last TCAM block in the cascade. Therefore, when using a cascade, the action memories (see Section 7.9.1.5.3) are associated with the last memory in the cascade.

Figure 7-21 illustrates an example of cascading two TCAM blocks to manifest a table with an 80-bit key. Using the configuration illustrated in the figure, a hit indication in the last TCAM (TCAM *n+1* in the figure) is only possible if the same entry is hit in both the last TCAM and the previous TCAM (TCAM *n* in the figure) because the StartCompare input for TCAM *n+1* is cleared. The first TCAM, however, ignores the previous TCAM block (TCAM *n-1* in the figure) hit indications, since it's StartCompare input is set.

Note: As highlighted in the figure, the first TCAM block's (TCAM *n* in the figure) priority logic is unused. and instead, the last TCAM block's (TCAM *n+1* in the figure) priority logic output reflects the cascade hit indication and hit index. Therefore, in the case described in this example, the action memories associated with the table implemented using this cascade is associated with TCAM *n+1*.

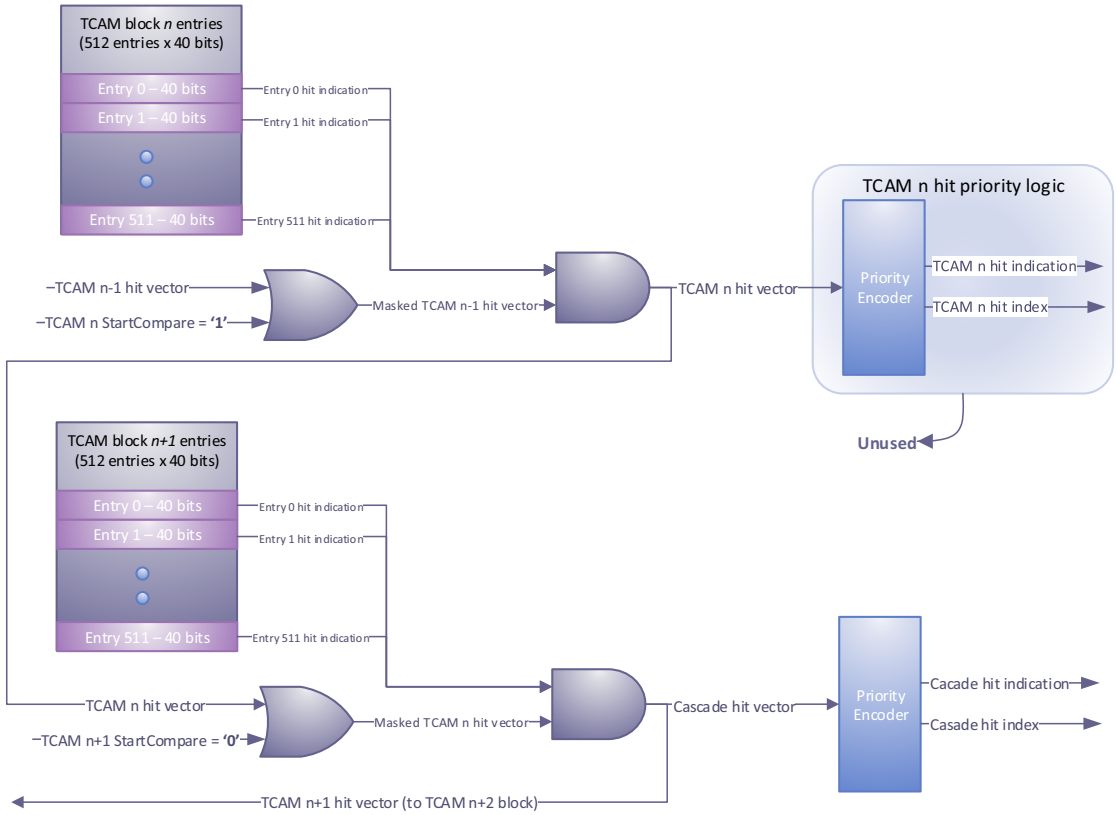


Figure 7-21. TCAM Cascade Example

7.9.1.4.2.2 TCAM Stacking

TCAM stacking is used to support tables with a key amount higher than 512. The TCAM stacking mechanism operates by running several TCAM blocks lookup in parallel and returning a unified result entry that represents the highest indexed entry hit in the highest indexed TCAM block in the stack. Figure 7-22 shows and example of two cascaded tables 1024x80b table and a 1024x40b table.

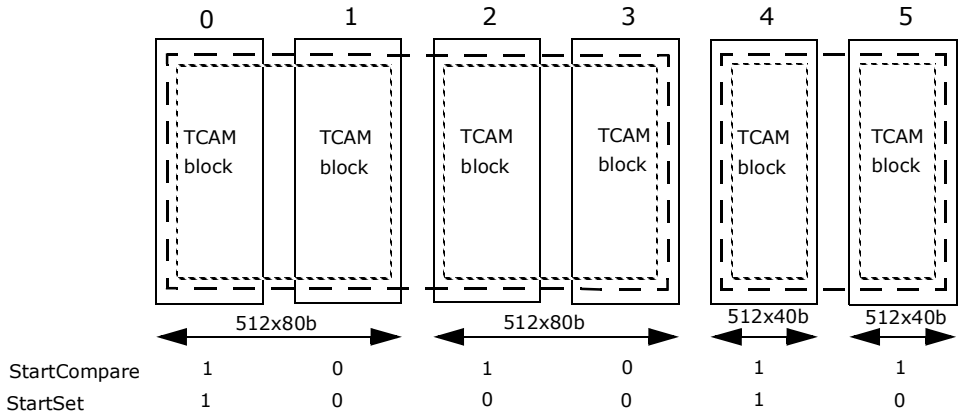


Figure 7-22. Cascaded TCAM Example

The ACL block requires stacked TCAM blocks to be adjacent one to another (that is, to have consecutive indexes). For example, a table supporting an 40-bit wide key and 1,024 entries can be implemented using TCAM blocks index 1 and 2, but not using TCAM block 1 and 3.

When stacked, the hit vector, representing the hit indications for an entire TCAM, disables the hit of the next TCAM. As such, only the first hit of the cascaded TCAM provides the result of the entire table.

Note: TCAM stacking and TCAM cascading can be used jointly to form. For example, a table with 1,024 entries and a key of 80 bits.

This setting is controlled by the StartSet configuration, which is part of the scenario configuration per TCAM (see Section 7.9.2.5.1). The action memory are associated with every TCAM that completes a table entry.

7.9.1.4.3 Concurrent Table Support

Concurrent tables are tables that are searched in parallel for the same input. An example of concurrent tables is a MAC Address table and an IPv4 Address table which can be searched concurrently for the same input. Figure 7-23 shows a concurrent TCAM example. The two tables can be searched concurrently since they occupies different TCAMs

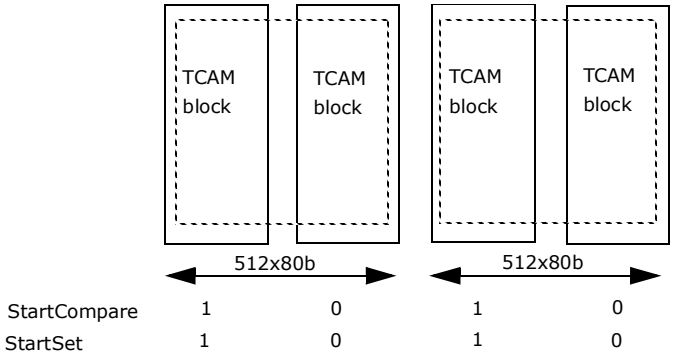


Figure 7-23. Concurrent Tables Example

The ACL block in the E810 supports multiple concurrent tables. This is done by using several TCAM blocks in parallel for implementing different tables that are searched in parallel for each input. Each table supports its own key structure configuration.

The amount of concurrent tables supported by the design is equal to the amount of TCAM blocks in the design (that is, support a case where each TCAM block is used to implement a single table).

Note: Concurrent tables usage is not related to TCAM cascading (see Section 7.9.1.4.2.1) or TCAM stacking (see Section 7.9.1.4.2.2). This means that some of the concurrent tables can use a stacked or cascaded TCAM configuration.

7.9.1.4.4 Non-Concurrent Table Support

The ACL block in the E810 supports multiple non-concurrent tables (or, tables that are not working concurrently). An example of non-concurrent tables is an inner IPv6 Address table and an inner IPv4 Address table that are naturally never searched concurrently. Figure 7-24 shows an example of non-concurrent tables. Because the two tables consume the same TCAM block they cannot be searched concurrently. In fact they are going to be define in a different scenarios.

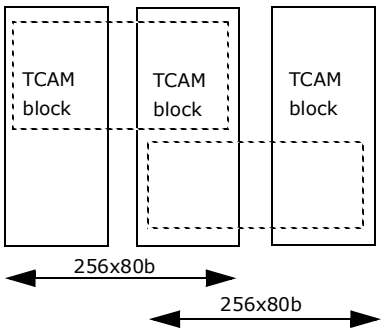


Figure 7-24. Non-Concurrent Tables Example

Supporting multiple non-concurrent tables is used to optimize the use of TCAM blocks when tables are not using the full amount of entries supported by a TCAM block used to implement it.

The ACL block in the E810 supports multiple non-concurrent tables by allowing the same TCAM block to store multiple tables while masking the hits in the irrelevant tables on a per-input basis. Entry masking is possible in 64-entries resolution.

Note: Though the example given above is implying using the packet type as a mean to select the relevant (or masking the irrelevant tables) per input, this feature can use other information (like the function ID) for selecting the active tables to allow different functionality (like function isolation).

7.9.1.5 Actions

7.9.1.5.1 General

If a table yielded a hit, the result entry points to a specific set of actions associated with the entry. The ACL block supports at least two actions per entry and up to 20,480 actions in total for all entries. Example actions might be: count bytes, count packets, drop, and change VSI.

The action format supported by the ACL block is the format supported by the flexible pipeline architecture and includes an action priority, action value and action metadata ID (MDID).

The ACL block supports up to 8 actions per a single output.

7.9.1.5.2 Aggregation and Prioritization

As mentioned in Section 7.9.1.3, the ACL block supports multiple concurrent tables working simultaneously. Potential multiple tables hit results in multiple actions being aggregated to a single set of actions.

A single set of ACL actions may include only a single action per action type (or, metadata ID). When two or more table hits result in the same action type, a priority mechanism uses the action priority, which is part of the action definition associated with each action, to determine which is used in the output. When two or more table hits result in the same action type with the same priority, the action from the highest index action memory is used in the output.

Note: Any combination of table hit that results in an action set of more that eight unique actions, corresponding to the ACL block limit mentioned in Section 7.9.1.5.1, is considered as an invalid programming and might result in unexpected behavior.

Similar to the aggregation and prioritization mechanism in the ACL block, the ACL block's actions are aggregated with and prioritized against the pipeline's actions. When a result action in the ACL block has the same type of a pipeline action, the highest priority action prevails. When two priorities of the same action type are equal, the ACL action prevails unless the priority is set to 0.

7.9.1.5.3 Action Memories

The actions are stored in arrays referred to as action memories. The ACL block in the E810 includes 20 action memories, where each memory has 512 entries containing up to two actions per entry.

Any action memory can be associated with any of the TCAM blocks so that a hit in a specific entry of the TCAM block points to the same entry in the associated action memory. This allows the action memories to support flexible table sizing as mentioned in Section 7.9.1.4.2. Several action memories can also be associated with the same TCAM block, allowing the support for more than two actions per table hit.

Figure 7-25 shows and example of action memory-to-TCAM block association. A single action memory can be associated with only on TCAM block per scenario.

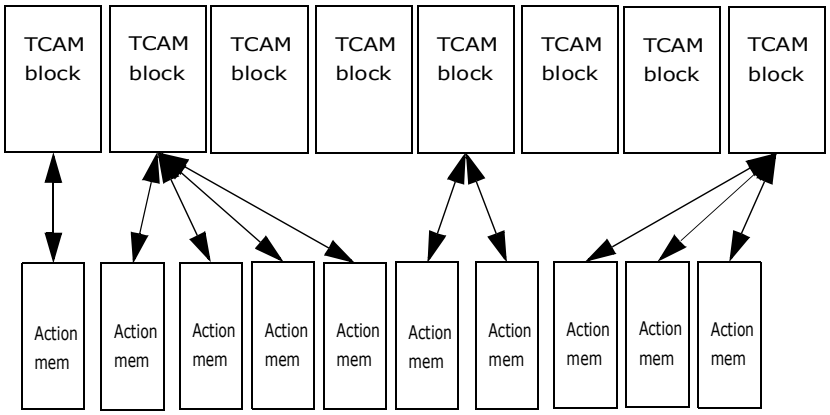


Figure 7-25. Action Memory-to-TCAM Association Example

7.9.1.5.4 Default Action

There are three special cases that involve associating an action with an incoming command when there is no valid action as a result of the ACL operation:

- Profile miss — An incoming command that did not select any profile (see Section 7.9.2.2) and was thus associated with by pass profile.
- Tables miss — An incoming command that did not match any entry in the ACL (that is, missed all tables it searched through).

3. Table hit without any valid action.

When any of these cases occurs, the output actions is defined by the default action mechanism. The default action mechanism allows associating each VSI with four independent default action sets. An independent action set can be selected for profile miss and table miss and for Rx and Tx traffic (see Section 7.9.2.8).

Each default action set contains up to four predefined actions. An example for a default action set might be: drop the packet, count packets, and set a special tag that is reflected in the Rx-Descriptor for software.

Note: It is not expected that there are two actions in the default set with the same MDID and different priority. However, if this happens, only one of them is selected, disregarding the other.

Note: A hit where all the action associated with it has priority 0 results the invocation of the default actions. To avoid that, the NOP action with non-0 priority value is used. Then, the hit causes no action.

7.9.1.5.5 VSI Reassignment

One of the actions supported by the ACL block is to change the destination VSI (applicable only for Rx packets). When this action is selected and based on its priority compared to previous set VSI actions, it causes the destination VSI associated with the packet to change.

When a VSI change occurs, the ACL block also updates the function context based on the new VSI value, according to the value in the applicable VSI_VSI2F CSR.

The following cases causes the ACL block to mark the packet to be dropped following a VSI reassignment:

- The newly assigned VSI is disabled (i.e., VSI_VSI2F[VSI].*ENABLE* is clear).
- The packet was an encrypted packet that went through in-line decryption. The newly assigned VSI is different than the original VSI, and GL_SWT_FUNCFILT.*FUNCFILT* is asserted.

7.9.1.6 Counters

7.9.1.6.1 General

The count action is a fundamental feature of the ACL block. One or more count actions can either be associated with an entry hit by programming it to the relevant entry in the action memory (see Section 7.9.1.5.3), or be associated with a default action for a table/profile miss (see Section 7.9.1.5.4).

Note: Counters are an action associated with an entry(ies). Therefore, counters are agnostic to the traffic type or properties, and can count, for example, Tx/Rx traffic in the same counter.

Note: In case VSI reassignment has occurred, the VSI used for checking the enablement for counting is the "new VSI" as defined in Section 7.9.1.5.5.

7.9.1.6.2 Counting Drops

Count actions might be part of an action set that includes a drop command (that is, one or more tables hit resulted in both count and drop actions).

Note: Counter always counts. This allows the ability to count the number of dropped packets.

7.9.1.6.3 Counter Banks

The ACL block in the E810 supports 2048, 40-bit wide, counters arranged in four banks/arrays, and is able to support up to four counter updates for the same search.

Each counter bank/array contains 512 counters according to the following distribution:

- Bank 0 contains counter indexes 0-511.
- Bank 1 contains counter indexes 512-1023.

- Bank 2 contains counter indexes 1024-1535.
- Bank 3 contains counter indexes 1536-2047.

Note: The update of more than a single counter in the same bank is forbidden and can yield unexpected behavior.

7.9.1.6.4 Counters Read and Reset

The ACL counters are accessible via the following CSRs, where "bank offset" is the offset of the counter withing the bank:

- Bank 0 counters are accessible via GLSTAT_ACL_CNT_0_L/H[bank offset].
- Bank 1 counters are accessible via GLSTAT_ACL_CNT_1_L/H[bank offset].
- Bank 2 counters are accessible via GLSTAT_ACL_CNT_2_L/H[bank offset].
- Bank 3 counters are accessible via GLSTAT_ACL_CNT_3_L/H[bank offset].

Since the counters are wider than a single 32-bit CSR access, the ACL block counters supports a 64-bit CSR access. A 64-bit access causes the hardware to latch the entire value of the counter and allows reading it using 2two consecutive 32-bit transactions without needing to check for wrap-around terms.

The counters are also accessible for write transaction. Any write transaction to a counter causes its value to be reset.

7.9.1.6.5 Supported Count Actions

The ACL block supports three different action commands/MDIDs (see Section 7.9.1.5.1) for manipulating counters:

- **Count packets (MDID 56)** — When this action is used, the counter referenced by the action's value field is incremented by one. The counter ID used in this command can be in the full range of counter indexes (0-2047) under the restriction on updates in the same bank (see Section 7.9.1.6.3).
- **Count bytes (MDID 57)** — When this action is used, the counter referenced by the action's value field is incremented by the amount of bytes contained in the packet (the packet length). The counter ID used in this command can be in the full range of counter indexes (0-2047) under the restriction on updates in the same bank (see Section 7.9.1.6.3).
- **Count packets and bytes (MDID 58)** — When this action is used, the action's value field references a counter dual and updates two counters in two separate banks. The counter referenced in the action's value field is incremented by the one (that is, serves as a packet counter), while the counter indexed (index in the command +1024) is incremented by the amount of bytes contained in the packet (in other words, serves as a byte counter). The counter ID used in this command can be in half the range of counter indexes (0-1023) under the restriction on updates in the same bank (see Section 7.9.1.6.3).

Given that the ACL block only allows a single action of each type (see Section 7.9.1.5.2) and the restriction on same bank counter updates (see Section 7.9.1.6.3), the amount of valid combinations of counter manipulation actions for a single search is limited and is summarized in Table 7-79.

Table 7-79. Counter Actions Possible Combinations

Counter Updates	Bank 0 (Index 0-511)	Bank 1 (Index 512-1023)	Bank 2 (Index 1023-1535)	Bank 3 (Index 1536-2047)
0	None	None	None	None
1	Count bytes or packets	None	None	None
1	None	Count bytes or packets	None	None
1	None	None	Count bytes or packets	None
1	None	None	None	Count bytes or packets
2	Count bytes	Count packets	None	None
2	Count packets	Count bytes	None	None
2	Count bytes	None	Count packets	None
2	Count packets	None	Count bytes	None
2	Count bytes	None	None	Count packets
2	Count packets	None	None	Count bytes
2	None	Count bytes	Count packets	None
2	None	Count packets	Count bytes	None
2	None	Count bytes	None	Count packets
2	None	Count packets	None	Count bytes
2	None	None	Count bytes	Count packets
2	None	None	Count packets	Count bytes
2	Count bytes and packets	None	None	None
2	None	Count bytes and packets	None	None
3	Count bytes and packets	Count bytes or packets	None	None
3	Count bytes or packets	Count bytes and packets	None	None
3	Count bytes and packets	None	None	Count bytes or packets
3	None	Count bytes and packets	Count bytes or packets	None
4	Count bytes and packets	Count bytes	None	Count packets
4	Count bytes and packets	Count packets	None	Count bytes
4	Count bytes	Count bytes and packets	Count packets	None
4	Count packets	Count bytes and packets	Count bytes	None

Note: When an action set contains two actions of the same type (for example, two “count bytes” actions), the hardware uses the priority mechanism to prioritize and yields a single action of each type, thus such programming is allowed (see Section 7.9.1.5.2). However, the hardware does not implement a mechanism to resolve multiple counter updates in the same counter bank.

7.9.1.7 Key Selection

7.9.1.7.1 General

The key used for each input depends on multiple factors including the packet type, the function, and the matched fields. It is therefore done in three separate levels:

1. Global level, selection base creation (see Section 7.9.1.7.2).
2. TCAM block level selection (see Section 7.9.1.7.3).

Each level is controlled by a different configuration. The three-level approach allows both flexible usage and optimized utilization of the TCAM blocks which are a relatively limited resource.

7.9.1.7.2 Selection Base

Figure 7-26 shows the generation of the selection base.

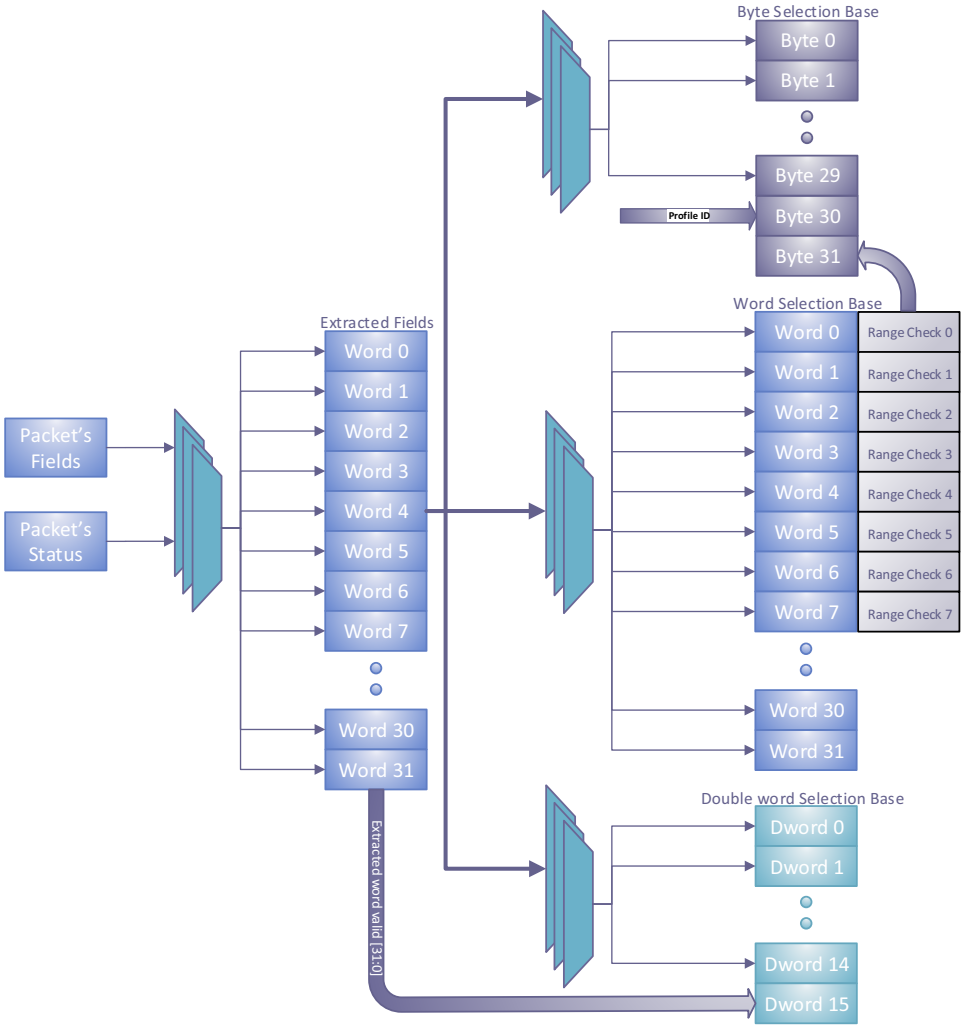


Figure 7-26. Selection Base Generation

7.9.1.7.2.1 Selection Base Vectors

The creation of the selection base involves the selection of the packet's properties and header fields that are available for all the TCAM blocks.

This stage is implemented using two sub-stages:

1. The extractor creates an extracted fields vector containing up to 32 words from the packet's header fields and status.
2. The key generation block selects fields from the extracted fields vector and generates the selection base.

Both extractor and key generation block configuration source is associated with a profile (see Section 7.9.2.2).

The selection base is comprised of the selection base vectors accumulating to 160 bytes (see Figure 7-26):

1. **Byte selection base** — A 32-byte vector containing byte-sized fields (32 fields supported). Selection of bytes from this vector for a match is done in byte resolution.
2. **Word selection base** — A 64-byte vector containing word-sized fields (32 fields supported). Selection of bytes from this vector for a match is done in word resolution.
3. **Double-word selection base** — A 64-byte vector containing double-word-sized fields (16 fields supported). Selection of bytes from this vector for a match is done in double-word resolution.

7.9.1.7.2.2 Range Checkers

A range checker is a mechanism that allows checking if a specific field, representing a natural number, is within range of two given natural numbers (that is, defining the boundaries).

The range checker output is positive if the field associated with it is within range (is higher than or equal to the low boundary and is lower than or equal to the high boundary), and is negative if it is out of the given range (is lower than the low boundary or is higher than the high boundary).

The ACL block in the E810 supports eight word-sized range checkers working concurrently. Each of the eight range checkers is associated with a specific word in the word selection base (see Figure 7-26). Also, each of the words mapped to a range checker includes a bit-level masking mechanism, which allows checking ranges for a sub-field in each word.

Note: Masking of words mapped to the range checkers is done between the word in the word selection base and the associated range checker, and is employed only for range checking purposes. This means that if a table selects one of these words as part of its key, it gets the unmasked version of the word.

The range checkers output is mapped to the byte selection base (see Section 7.9.1.7.2) and can be used for matching.

The range checkers boundaries are a profile dependent configuration (see Section 7.9.2.6), and thus the ACL block can virtually support 1,024 range checkers (eight range checkers per profile).

Note: Logical combination of several checkers for forming a single match is possible. For example, by requiring several range checkers positive/negative output in the relevant table entry a logical AND can be created between range checkers.

7.9.1.7.2.3 Fixed Fields and Restrictions

The selection of fields to each selection base vector is part of the profile specific programming (see Section 7.9.2.2) and is (in general) flexible, meaning that any field can be selected to any location in the selection base. There are, however, several fixed fields located in the selection base, which impose restrictions that apply for selection of fields to the selection base.

Byte selection base fixed fields:

- Byte index 31 contains the range checkers output (see Section 7.9.1.7.2.2). Each bit index *n* in this byte corresponds to a range checker index *n* output.
- Byte index 30 contains the selected Profile ID.

Word selection base restrictions:

- Word indexes 0-7 are the only words that can be used for range check (see Section 7.9.1.7.2.2). Each word index *n* is associated with range checker index *n* input.

Note: It is emphasized that the restriction mentioned above, only relates to the ability of selecting a source for the range checkers. This does not impose any restriction on the ability of selecting a word from the extracted fields to these words in the word selection base.

Double-word selection base fixed fields:

- Double-word index 15 contains the extracted fields vector valid indication per word (a total of 32 valid indication bits). Each word *n* of the extracted fields vector valid indication is associated with bit *n* in double-word 15.

7.9.1.7.3 TCAM Key Byte Selection

The TCAM key byte selection occur in each TCAM level. As mentioned in Section 7.9.1.4.2, the ACL block in the E810 includes 16 TCAM blocks, each supporting a 40-bit wide key. Each TCAM block supports selecting each of the five bytes of the key used for a match out of the three fields vectors constituting the selection base (see Section 7.9.1.7.2).

The configuration of the TCAM byte selection is associated with a scenario (see Section 7.9.1.8) and depends on the structure of the selection base, the tables being used (that is, the fields being matched) and the specific table configuration used to implement the tables (see Section 7.9.1.4). The same factors are taken into account when programming the contents of each TCAM block.

The TCAM byte selection available options are defined in Table 7-80, where each byte is selected independently.

Table 7-80. TCAM Byte Selection Options

Byte Index	Bit Indexes	Byte Fields	Word Fields	Double Word Fields
4	[39:32]	Any byte (32 options)	Not available	Not available
3	[31:24]	Any byte (32 options)	Bits 15:8 of any word (32 options)	Bits 31:24 of any double word (16 options)
2	[23:16]	Any byte (32 options)	Bits 7:0 of any word (32 options)	Bits 23:16 of any double word (16 options)
1	[15: 8]	Any byte (32 options)	Bits 15:8 of any word (32 options)	Bits 15:8 of any double word (16 options)
0	[7: 0]	Any byte (32 options)	Bits 7:0 of any word (32 options)	Bits 7:0 of any double word (16 options)

7.9.1.8 The Scenario Mechanism

A scenario is a set of configuration including:

- TCAM blocks configuration (cascading or stacking) for supporting flexible table sizing (see Section 7.9.1.4.2).
- TCAM blocks enablement for supporting concurrent table support (see Section 7.9.1.4.3).
- TCAM blocks specific entries enablement for supporting non-concurrent table support (see Section 7.9.1.4.4).
- Key selection configuration for each TCAM block (see Section 7.9.1.7.3).
- Association of action memories to TCAM blocks and enablement of each action memory (see Section 7.9.1.5.3).

The ACL block in the E810 supports 64 independent scenario sets. The selection of which scenario set to use for a given input is based on the profile selected (see Section 7.9.2.6) and the PF.

7.9.2 ACL Programming

7.9.2.1 General

This section depicts the relevant flows and methods for programming the ACL block.

7.9.2.2 Configuration Sources

There are three types of configuration origins in the ACL block (see Figure 7-27):

- **Global Configuration** — A block-wide configuration that is not dependent on the incoming command.
- **Profile-Based Configuration** — Settings are loaded according to the selected Profile ID. Each of these settings has a unique value for each Profile ID.
- **Scenario-Based Configuration** — Settings are loaded according to the selected scenario. Each of these settings has a unique value for each scenario.

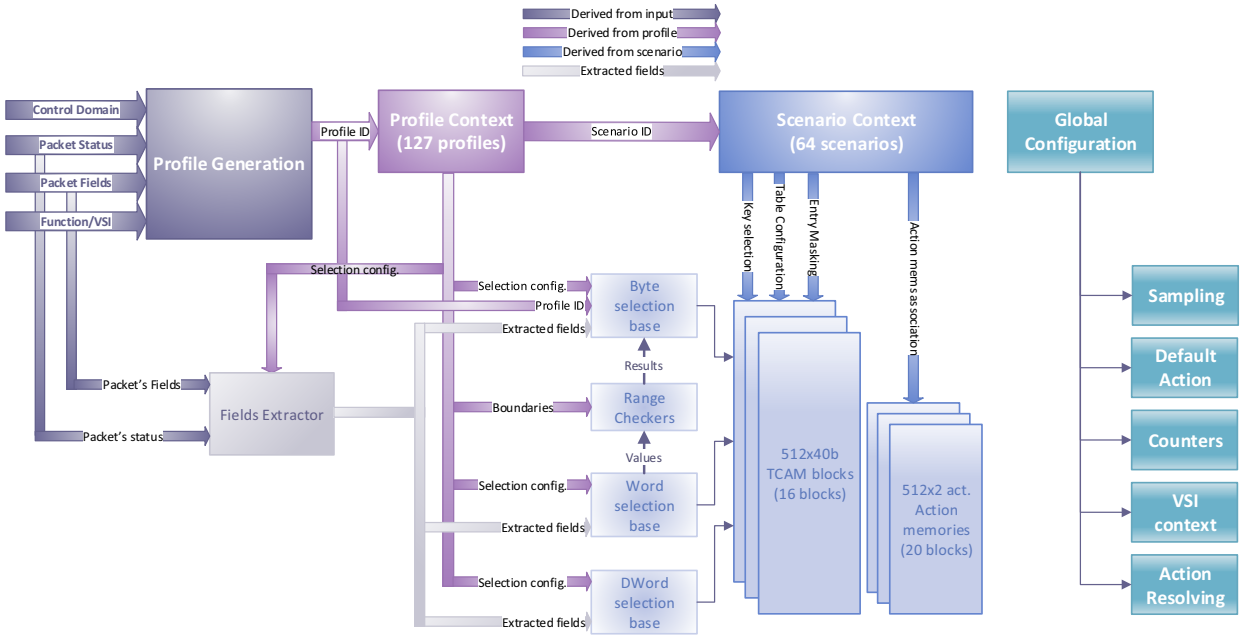


Figure 7-27. Configuration Sources[1]

1. Number of profiles is 127 to signify that one profile is the bypass one.

7.9.2.3 TCAM Entry Configuration

As described in Section 7.9.1.4, a TCAM block is 40 bits wide and can be used to implement a table or part of a table. No matter which of part of the table the TCAM block is implementing, its entry is configured using the same basic principals.

The programming of the TCAM blocks' entries is done using the indirect access mechanism. Each TCAM block entry programming is comprised of writing two values:

- **The key** — A 40-bit wide value that is matched against an incoming key/key part. This value is programmed using the GL_ACL_TCAM_KEY_L/H CSRs.
- **The key invert** - A 40-bit wide 1's compliment value of the key used to define a "don't care" or a mask operation (see Section 7.9.1.3). This value is programmed using the GL_ACL_TCAM_KEY_INV_L/H CSRs.

The functionality per entry bit derived from the different combinations of key and key invert programming is explained in Table 7-81.

Table 7-81. Key and Key Invert Programming Effect

Key Value	Key Invert Value	Functionality
1	1	Don't care (bit always matches)
1	0	Entry bit holds a '0' value.
0	1	Entry bit holds a '1' value.
0	0	Bit never matches (invalidate entry)

7.9.2.4 Action Memory Entry Configuration

As described in Section 7.9.1.5.3, the ACL block includes 20 action memories, each containing 512 entries of two actions.

The programming of action memories entries is done using the indirect access mechanism. The two actions associated with each entry is programmed using the GL_ACL_ACTMEM_CFG array.

- The GL_ACL_ACTMEM_CFG[action index].*PRIORITY* field is programmed with the action's priority to be used for the action prioritization mechanism (see Section 7.9.1.5.2).
- The GL_ACL_ACTMEM_CFG[action index].*MDID* field is programmed with the action's metadata identifier.
- The GL_ACL_ACTMEM_CFG[action index].*VALUE* field is programmed with the action's value (the meaning of the value depends on the action type).

7.9.2.5 Scenario-Dependent Configuration

As described in Section 7.9.1.8, the ACL block supports a total of 64 scenarios.

The programming of the scenario dependent configuration is done using the indirect access mechanism.

The configuration associated with each scenario is divided to two configuration sets (see Figure 7-28):

- Scenario Configuration per TCAM block (see Section 7.9.2.5.1).
- Action memories association to TCAM blocks (see Section 7.9.2.5.2).

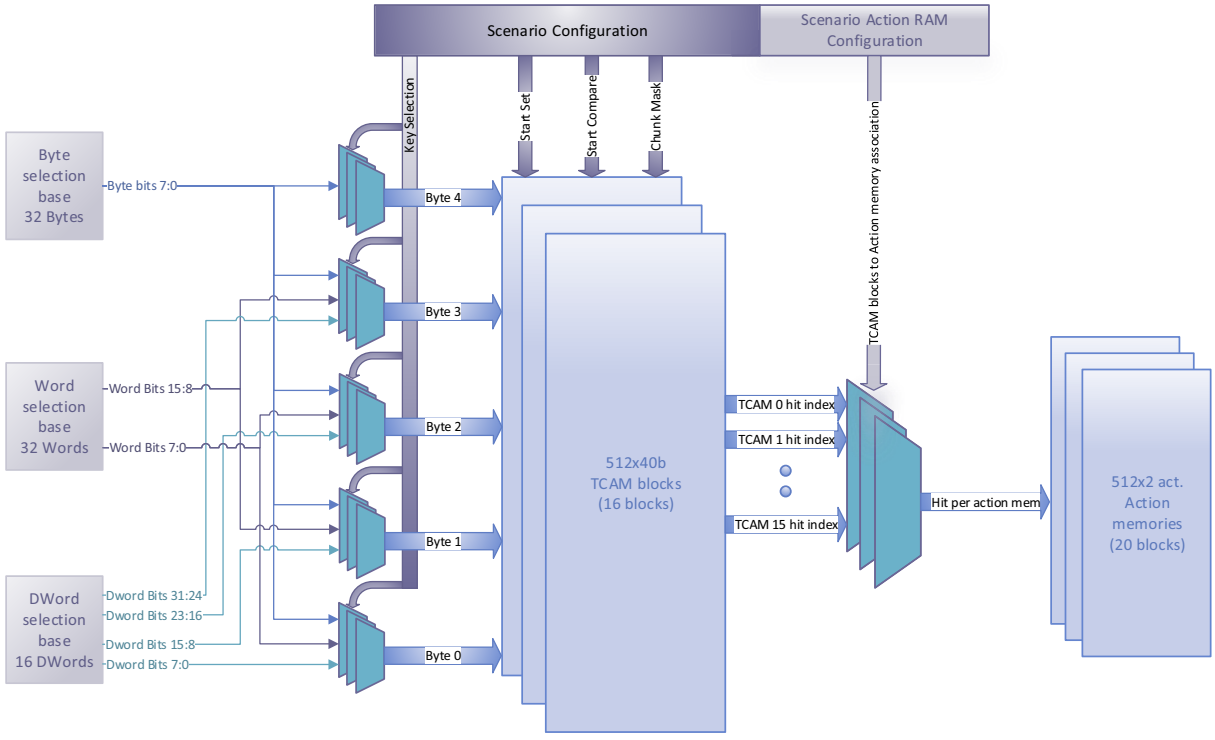


Figure 7-28. Scenario-Dependent Configuration

7.9.2.5.1 Scenario Configuration per TCAM

Scenario Configuration per TCAM block includes:

- Key selection for each of the 5-byte key for each TCAM (see Section 7.9.1.7.3):
  - Byte 0 selection is programmed in GL_ACL_SCENARIO_CFG_L[TCAM index].*SELECT0*.
  - Byte 1 selection is programmed in GL_ACL_SCENARIO_CFG_L[TCAM index].*SELECT1*.
  - Byte 2 selection is programmed in GL_ACL_SCENARIO_CFG_L[TCAM index].*SELECT2*.
  - Byte 3 selection is programmed in GL_ACL_SCENARIO_CFG_L[TCAM index].*SELECT3*.
  - Byte 4 selection is programmed in GL_ACL_SCENARIO_CFG_H[TCAM index].*SELECT4*.
  - See CSR fields description for more information about the selection value.

  Note: Refer to Table 7-80 for the selection options.
- Entry masking in 64-entries resolution for each TCAM (see Section 7.9.1.4):
  - The masking is programmed in GL_ACL_SCENARIO_CFG_H[TCAM index].*CHUNKMASK*.
  - Each bit masks 64 entries (that is, bit 0 masks entries 0-63, bit 1 masks entries 64-127, and so on).

  Note: When an entire TCAM is masked (CHUNKMASK for a specific TCAM is set to 0x00), the TCAM block is power gated.
- TCAM blocks configuration for implementing flexible table sizes (see Section 7.9.1.4.2):
  - The StartCompare setting is programmed in GL_ACL_SCENARIO_CFG_H[TCAM index].*START_COMPARE*.
  - The StartSet setting is programmed in GL_ACL_SCENARIO_CFG_H[TCAM index].*START_SET*.

7.9.2.5.2 Action Memories-to-TCAM Association

The mapping of action memories to TCAMs is programmed using the indirect access mechanism.

Each memory can be associated to a single TCAM block, thus making it possible to support a topology in which the same TCAM block is associated with several action memories (to support more than two actions per hit) or be completely disabled (allowing it to be power-gated). When using TCAM cascade (see Section 7.9.1.4.2.1), the action memory(ies) associated with the cascade is associated with the last TCAM block in the cascade.

The TCAM block index associated with the action memory is programmed in GL_ACL_SCENARIO_ACT_CFG[action memory index].*ACTMEM_SEL*.

Action memory enable/disable status is programmed in GL_ACL_SCENARIO_ACT_CFG[action memory index].*ACTMEM_EN*.

7.9.2.6 General Profile-Dependent Configuration

Any incoming command passes through the profile selection stage as the initial stage in the ACL. The selection of the profile is similar to all blocks in the packet processing pipeline and depends on the packet's properties and associated function.

The profile selection result is the Profile ID which loads a specific profile dependent configuration which is unique per profile. The profile dependent configuration in the ACL block includes (see Figure 7-27):

- **Fields extractor selection** (see Section 7.5).
- **Byte selectionbase select** — Selection for each of the bytes in the byte selection base programmed in GL_ACL_PROFILE_BWSB_SEL[byte index].*BSB_SRC_OFF*.
- **Word selectionbase select** — Selection for each of the words in the words selection base programmed in GL_ACL_PROFILE_BWSB_SEL[word index].*WSB_SRC_OFF*.
- **Double-word selection base select** — Selection for each of the double-words in the double-words fields vector programmed in GL_ACL_PROFILE_DWSB_SEL[DWord index].*DWORD_SEL_OFF*.

Scenario selection per PF programmed in GL_ACL_PROFILE_PF_CFG[PF index].*SCEN_SEL*.
Profile-dependent configuration access is done using the indirect access mechanism.

Note: As mentioned in Section 7.9.1.5.4, profile index 0 is a special profile index reserved for input commands that are not associated with any profile and are thus not performing any search in the ACL. Attempting to program a profile dependent configuration for profile index 0 is treated by the hardware as an error.

Note: As mentioned in Section 7.9.1.7.2.3, bytes indexes 30 and 31 in the byte selection base are occupied by fixed fields and thus the value of GL_ACL_PROFILE_BWSB_SEL[30].*BSB_SRC_OFF* and GL_ACL_PROFILE_BWSB_SEL[31].*BSB_SRC_OFF* is ignored by hardware.

7.9.2.7 Range Checkers Profile-Dependent Configuration

As mentioned in Section 7.9.1.7.2.2, the range checkers configuration is a profile-dependent configuration. The range checkers profile dependent configuration in the ACL block includes (see Figure 7-27):

- Range checkers boundaries for each of the range checkers programmed in GL_ACL_PROFILE_RC_CFG[range checker index].
- Range checkers fields mask for each of the range checkers programmed in GL_ACL_PROFILE_RCF_MASK[range checker index].

The range checkers profile dependent configuration access is done using the indirect access mechanism.

7.9.2.8 Default Action Programming

As described in Section 7.9.1.5.4, the ACL block allows each VSI to be associated with two sets of actions (out of possible four) that are used as the action set in case of a profile miss or a table miss.

The action sets are configured in the GL_ACL_DEFAULT_ACT array in which every array member defines a single action (out of possible four) in a single set (out of possible four).

- GL_ACL_DEFAULT_ACT[set index*4action index].*PRIORITY* is used to program the action's priority.
- GL_ACL_DEFAULT_ACT[set index*4+action index].*MDID* is used to program the action's Metadata ID.
- GL_ACL_DEFAULT_ACT[set index*4+action index].*VALUE* is used to program the action's value.

For an Rx input, the selection of default action set is programmed for a destination VSI (as determined by a previous block such as the switch) in VSI_ACL_DEF_SEL[VSI index].*RX_PROFILE_MISS_SEL* for profile miss, and in VSI_ACL_DEF_SEL[VSI index].*RX_TABLES_MISS_SEL* for tables miss.

For a Tx input, the selection of default action set is programmed for a source VSI in VSI_ACL_DEF_SEL[VSI index].*RX_PROFILE_MISS_SEL* for profile miss, and in VSI_ACL_DEF_SEL[VSI index].*RX_TABLES_MISS_SEL* for tables miss.

Note: If one or more of the default actions involves counter manipulation, all the rules and restrictions related to counter manipulation is followed (see Section 7.9.1.6).

7.9.2.9 VSI Count Programming

Count enablement per VSI (see Section 7.9.1.6.1) is programmed in VSI_ACL_CNT_EN[VSI].*COUNT_EN*.

Note: In case of a VSI reassignment, the VSI referred to with respect to VSI count enablement is be the new VSI.

7.9.2.10 Reset and Initialization

7.9.2.10.1 Reset Sources

The ACL block's main reset source is the device's core reset (CORER) or any higher level reset source that triggers a core reset. When CORER is asserted, the hardware clears:

- Profile association and profile dependent configuration (see Section 7.9.2.6).
- Scenario configuration per TCAM (see Section 7.9.2.5.1).
- Action memories to TCAM association (see Section 7.9.2.5.2).
- Counters (see Section 7.9.1.6).
- Default action configuration (see Section 7.9.2.8).

Hardware does not handle function resets. It is up to the firmware or software to handle these resets.

When a PF is being reset, firmware releases the resources allocated to it (see Section 7.9.3.2), which makes the resources available and guarantees that no incoming command searches the reseted PF tables. It is emphasized that the firmware is not required to invalidate the table entries contents.

Note: When a graceful PF reset occurs (that is, the software driver unloads), the software driver is expected to release its resources in a graceful manner (see Section 7.9.3.2) by issuing a command to the firmware to release all its resources (see Section 7.9.3.4.2.9).

When a VF is being reset, software (the relevant PF driver) clears (invalidate) the entries associated with the VF being reset.

7.9.2.10.2 Initialization

7.9.2.10.2.1 Initial State and Reset Values

With the exception of the default action sets (see Section 7.9.1.5.4) that are expected to be loaded by either NVM or firmware, the ACL block has no pre-configured content following a hardware reset event (see Section 7.9.2.10.1). This means that there will not be any profiles allocated or associated with any traffic type and hence no scenarios are allocated nor tables populated. All incoming traffic is expected to hit the default "no profile hit" profile (programmed in GL_ACLEXT_DFLT_L2PRFL_ACL).

The only applicable flow at this state is the default action flow related to profile miss (see Section 7.9.1.5.4). This flow is only relevant once VSIs are added to the system and thus, when adding a VSI to the system, the default action set selection for that VSI must be programmed (see Section 7.9.2.8) as part of the process.

7.9.2.10.2.2 Allocated Resources Initialization

No assumption is made on the initial content of allocated resources (see Section 7.9.3.2). Whenever a resource is allocated, the software driver that received the resource allocation completely initializes/ programs the resource.

Specifically, no assumption is made regarding the contents of the allocated table entries (see Section 7.9.3.2.1) and software is expected to invalidate all allocated entries prior to enabling the table entries in the scenario.

As mentioned in Section 7.9.2.10.2.1, when a PF reset occurs, the EMP firmware releases the relevant function's allocated resources but does not clear/invalidate the contents.

7.9.3 ACL Operation and Management

7.9.3.1 General

This section depicts the concepts and flows related to the operation and management of the ACL block.

7.9.3.2 Resource Allocation

The ACL block's resources are shared by the system's functions. To allow efficient use of the ACL block's resources, the EMP firmware is used for managing the ACL block resources allocation.

When a PF driver wishes to populate any of the resources in the list below, it requests an allocation from EMP firmware using the relevant AQ command. Similarly, when a PF driver wishes to release an allocated resource it requests the EMP firmware to deallocate the relevant resources and allow other PFs to use it.

Note: When a PF is reset, the EMP firmware releases its resources without requiring the PF to issue the relevant AQ command/s.

The ACL block resources allocated by the EMP firmware are:

- TCAM blocks and entries
- Action memories
- Profiles
- Scenarios
- Counters

7.9.3.2.1 Tables and Actions Allocation and Programming

The driver is able to request a table allocation and a set of corresponding action memory entries in the ACL block using the allocate_acl_table AQ command (see Section 7.9.3.4.2.1). The EMP firmware allocates TCAM blocks and entries according to the rules and restrictions described in Section 7.9.1.4.

When allocating a new table, the software indicates whether the table is an non-concurrent table (see Section 7.9.1.4.4), which can be allocated in any TCAM block unrelated to other allocated tables, or a concurrent table (see Section 7.9.1.4.3), and then the software can specify up to 12 previously allocated tables which cannot share a TCAM block with the new table.

The allocated tables is rounded to 64x40-bit units. For example, when the driver requests to allocate a table of 120 entries of 60 bits, it allocates a 128x80-bit table (that is, two blocks of 64 entries in two adjacent TCAM blocks).

In addition, the EMP firmware allocates action memories according to the amount of actions requested in the command. Action memories are allocated according to the guidelines described in Section 7.9.1.5.3.

Note: Action pairs can allocated and added to the allocated table in a later stage using the allocate_acl_actionpair command (see Section 7.9.3.4.2.3). The minimal allocated action pairs (a set of two actions) per table is 1.

Similarly, the driver is able to request releasing a table allocated to it using the deallocate_acl_table AQ command (see Section 7.9.3.4.2.2). Following this command, the EMP firmware is allowed to allocate the TCAM blocks and entries in addition to the action memories associated with the released table to other drivers usage.

Note: An action pair that was allocated and added to an allocated table using the allocate_acl_actionpair command (see Section 7.9.3.4.2.3) can be deallocated using the deallocate_acl_actionpair command (see Section 7.9.3.4.2.4) without deallocating the entire table resources.

Table entry programming is done using the program_acl_entry command (see Section 7.9.3.4.3.5) and querying an entry is done using the query_acl_entry command (see Section 7.9.3.4.4.4). When programming or querying a table entry, the EMP firmware follows the guidelines described in Section 7.9.2.3.

Action pairs can be programmed using the program_acl_actionpair command (see Section 7.9.3.4.3.2) and querying an action pair is done using the query_acl_actionpair command (see Section 7.9.3.4.3.5). When programming or querying an action pairs, the EMP firmware follows the guidelines described in Section 7.9.2.4.

7.9.3.2.2 Profile Allocation and ACL-Related Configuration

Profile allocation and deallocation in the ACL block is similar to the methods defined in other blocks (for example, the switch or post-switch) defined in Section 7.8.10, "Resource Allocation", and are beyond the scope of this discussion.

When a profile is requested by the driver, the EMP firmware allocates a free Profile ID and associates the profile with the requesting PF. It then allows the PF to program the general and range checkers profile-dependent configuration for the allocated Profile ID (see Section 7.9.2.6 and Section 7.9.2.7).

General profile-dependent configuration is done using the program_acl_profile_extraction command (see Section 7.9.3.4.3.3) and querying a general profile dependent configuration is done using the query_acl_profile command (see Section 7.9.3.4.4.1). When programming or querying a general profile-dependent configuration, the EMP firmware follows the guidelines described in Section 7.9.2.6.

Range checkers profile-dependent configuration is done using the program_acl_profile_ranges command (see Section 7.9.3.4.3.4) and querying a range checker profile dependent configuration is done using query_acl_profile_ranges command (see Section 7.9.3.4.4.2). When programming or querying a range checkers profile dependent configuration, the EMP firmware follows the guidelines described in Section 7.9.2.7.

7.9.3.2.3 Scenario Allocation and Programming

The driver is able to request a scenario allocation in the ACL block using the allocate_acl_scenario AQ command (see Section 7.9.3.4.2.5). When allocating a scenario, the software is able to program it with an initial programming.

The EMP firmware allocates a free scenario for the software and programs it with the initial programming requested in the command. Updating the configuration is possible using the update_acl_scenario command (see Section 7.9.3.4.3.1) and querying an allocated scenario configuration is done using the query_acl_scenario command (see Section 7.9.3.4.2.6).

When configuring/updating/querying a scenario, the EMP firmware follows the guidelines described in Section 7.9.2.5.

Similarly, the driver can request to deallocate a scenario allocated for it using the deallocate_acl_scenario command (see Section 7.9.3.4.2.6).

7.9.3.2.4 Counters Allocation

The driver is able to request an allocation of a block of counters using the allocate_acl_counters command (see Section 7.9.3.4.2.7). The counters allocated are of the same type (see more information about counter types in Section 7.9.1.6).

Similarly, the driver can request to deallocate a group of counters previously allocated for it using the deallocate_acl_counters command (see Section 7.9.3.4.2.8).

Note: The driver is not required to deallocate an entire block of counters allocated for it using the allocate_acl_command and can elect to deallocate only part of the counters allocated for it.

When allocating counters, the EMP firmware attempts to allocate a contiguous block of counters according to the driver's request. If the allocation fails, the driver might be able to allocate the same amount of counter in a non-contiguous block by invoking several allocation requests for smaller amounts of counters.

Note: The driver is responsible on maintaining the VSI related counters configuration (Section 7.9.2.9).

7.9.3.3 Software Flows

7.9.3.3.1 Initialization

As mentioned in Section 7.9.2.10.2, the ACL block default state is uninitialized. This means that even if software does not wish to use the ACL block, it has to define the default set of actions that is applied for each added VSI (see Section 7.9.1.5.4 and Section 7.9.2.8).

The configuration of the default action set used by the VSI for a profile miss and table miss is done using the Add VSI and Update VSI commands (see Table 7-41 for buffer format).

As there is no default configuration associated with the ACL block, all traffic that misses the profiles lookup, should be associated with profile 0 and be applied with the default action set associated with the VSI.

7.9.3.3.2 General Table Allocation Flow

As a prerequisite, before allocating and populating a table (and for each allocated table), software needs to determine the following items:

- Which header fields is available for lookup and range checking.
- What is the fields location in the selection base (byte/words/double words).
- The desired key structure that is used for lookup.

When the software wishes to allocate and populate a table in the ACL block, it executes the following flow:

1. Request a table allocation using the allocate_acl_table command (see Section 7.9.3.4.2.1). The table can be either concurrent or non-concurrent (see Section 7.9.3.2.1). The table width corresponds to the determined key width and the table depth corresponds to the expected amount of entries that is populated.

   Note: If the amount of action pairs associated with the table is known at this stage, software specifies it in the command. Otherwise, software specifies 1 (the minimal value) as the requested amount of action pairs to be associated with the table.

2. Initialize the resources allocated in Step 1:
   a. Invalidate all allocated TCAM entries by setting all key and key invert bits to each entry using the program_acl_entry command (see Section 7.9.3.4.3.5).
   b. Invalidate all allocated action pairs by clearing the action entries using the program_acl_actionpair command (see Section 7.9.3.4.3.2).
3. Allocate a scenario using the allocate_acl_scenario command (see Section 7.9.3.4.2.5) or alternatively, use a previously-allocated scenario and configure it to use the resources allocated in Step 1:
   a. Configure which TCAM blocks and entries are masked for this scenario.

      Note: Any TCAM block and entries that were not allocated for the driver must be masked.

   b. Configure the configuration of TCAM blocks (for example. cascaded, stacked).
   c. Configure the selection of key bytes into each of the TCAM blocks from the selection base vectors.
   d. Configure the association of action memories to TCAM blocks.
4. If no relevant profile was previously allocated, allocate a profile and configure the extracted fields vector.
5. Configure the ACL profile dependent configuration for the profile allocated/used in Step 4:
   a. Program the byte, word, and double-word selection bases composition by configuring the relevant selections.
   b. Program the scenario ID allocated in Step 3 to be associated with this profile.
6. Associate the ACL profile allocated/used in Step 4 with the relevant traffic/VSI.

   Note: At this stage, the table is allocated and a lookup is being performed for each relevant incoming packet. The software can start adding entries to the table (see Section 7.9.3.3.4) or action pairs to it (see Section 7.9.3.3.3).

7.9.3.3.3 Action Pair Allocation Flow

A prerequisite for allocating an action pair is to allocate a table using the table allocation flow (see Section 7.9.3.3.2).

When software wants to add another action pair to an existing table, it executes the following flow:

1. Request an action pair allocation using the allocate_acl_actionpair command (see Section 7.9.3.4.2.3).
2. Initialize the resources allocated in Step 1:
   a. Invalidate all allocated action pairs by clearing the action entries using the program_acl_actionpair command (see Section 7.9.3.4.3.2).
3. Update the relevant scenario using the update_acl_scenario command (see Section 7.9.3.4.3.1):
   a. Configure the association of allocated action memories to TCAM blocks.

   Note: At this stage. the action pair is allocated and is associated with the relevant tables. The software can start adding actions to it by invoking the program_acl_actionpair command (see Section 7.9.3.4.3.2).

7.9.3.3.4 Table Entries Addition Flow

A prerequisite for adding an entry to a table is to allocate a table using the table allocation flow (see Section 7.9.3.3.2).

When software wants to add an entry to a table, it executes the following flow:

1. Optionally, if used by the entry, configure the range checkers that is used in the profiles enabling this table using the program_acl_profile_ranges command (see Section 7.9.3.4.3.4).
2. Optionally, if used by the entry, allocate counter(s) for the entry using the allocate_acl_counters command (see Section 7.9.3.4.2.7).
3. Program the action pairs associated with the table (either by the action pair allocation flow or the table allocation flow) using the program_acl_actionpair command (see Section 7.9.3.4.3.4).
4. Program the entry key and key invert to the relevant entry using the program_acl_entry command (see Section 7.9.3.4.3.5) for all TCAM blocks forming the table.

   Note: At this stage, the entry is programmed in the table and any packet hitting it invokes the actions associated with it.

7.9.3.3.5 Table Entries Removal Flow

When software wants to remove an entry from a table, it executes the following flow:

1. Invalidate the relevant entry by setting all bits in the key and key invert using the program_acl_entry command (see Section 7.9.3.4.3.5) for all TCAM blocks forming the table.

   Note: At this stage, no packet hits the entry.
2. Optionally, if counter(s) were used by the entry and are no longer required, deallocate the counter(s) used by the entry using the deallocate_acl_counters command (see Section 7.9.3.4.2.8).
3. Optionally, if additional action pairs were added to the table using the allocate_acl_actionpair command (see Section 7.9.3.4.2.3) and are no longer required, deallocate the additional action pairs by using the deallocate_acl_actionpair command (see Section 7.9.3.4.2.4).

Note: At this stage, the entry is removed and all its private resources free.

7.9.3.3.6 Table Removal Flow

When software wants to remove an entire table, it executes the following flow:

1. If there are active scenarios that are enabling the table, mask the relevant TCAM entries/slices in all the scenarios that are enabling the table by using the update_acl_scenario command (see Section 7.9.3.4.3.1).

   Note: At this stage, no packet performs a lookup in the table.

2. Optionally, if counter(s) were used by the table and are no longer required, deallocate the counter(s) used by the table using the deallocate_acl_counters command (see Section 7.9.3.4.2.8).

3. Deallocate the table using the deallocate_acl_table command (see Section 7.9.3.4.2.2).

   Note: At this stage, the table is removed and all its private resources freed.

7.9.3.3.7 Scenario Removal Flow

Important: An active profile (a profile associated with active traffic) must point to a valid scenario. Therefore, the profile must be either disabled (for example, by preventing the relevant PF from selecting it), removed, or routed to a different valid scenario prior to removing the scenario.

When software wants to remove a scenario, it executes the following flow:

1. Prevent all profiles selecting the scenario from selecting the scenario by either:

   a. Preventing the PFs selecting the scenario from selecting the relevant profiles (unsubscribe the relevant VSIs from the profiles).

   b. Remove the profiles.

   c. Change the scenario pointed by the profiles to a different valid scenario by using the program_acl_profile_extraction command (see Section 7.9.3.4.3.3).

2. Deallocate the scenario by using the deallocate_acl_scenario command (see Section 7.9.3.4.2.6).

   Note: At this stage, the scenario is free.

7.9.3.4 ACL Block Admin Queue Commands

7.9.3.4.1 General

This section describes the Admin Queue commands related to the ACL operation and management. The Admin Queue operation is described in Section 9.5, and the general Admin Queue command format is described in Section 9.5.5.

The ACL block's Admin Queue commands are divided into three main groups:

- **Resource allocation commands** — These commands are used for resource allocation/ deallocation and, in some cases, initial programming or configuration of the resources.
- **Programming/update commands** — These commands are used for allocated resources programming or updating.

- **Query commands** — These commands are used for querying the allocated resources programming/status for reprogramming or debug.

The ACL block Admin Queue commands are summarized in Table 7-82.

Table 7-82. Admin Queue Commands Summary

Resource	Allocation/Deallocation	Programming	Querying
Tables	• allocate_acl_table Command (Section 7.9.3.4.2.1) • deallocate_acl_table Command (Section 7.9.3.4.2.2)	• program_acl_entry Command (Section 7.9.3.4.3.5)	• query_acl_entry Command (Section 7.9.3.4.4.4)
Actions	• allocate_acl_actionpair Command (Section 7.9.3.4.2.3) • deallocate_acl_actionpair Command (Section 7.9.3.4.2.4) ***Note:*** Action pairs are also allocated/deallocated as part of the table allocation.	• program_acl_actionpair Command (Section 7.9.3.4.3.2)	• query_acl_actionpair Command (Section 7.9.3.4.4.5)
Scenarios	• allocate_acl_scenario Command (Section 7.9.3.4.2.5) • deallocate_acl_scenario Command (Section 7.9.3.4.2.6)	• update_acl_scenario Command (Section 7.9.3.4.3.1) ***Note:*** The initial scenario configuration is done on allocation.	• query_acl_scenario Command (Section 7.9.3.4.4.3)
Profiles	• Profile allocation, Section 7.9.3.4.2.10, "Generic Allocate Resource Command" • Profile deallocation, Section 7.9.3.4.2.11, "Generic Free Resource Command"	• program_acl_profile_extraction Command (Section 7.9.3.4.3.3)	• query_acl_profile Command (Section 7.9.3.4.4.1)
Counters	• allocate_acl_counters Command (Section 7.9.3.4.2.7) • deallocate_acl_counters Command (Section 7.9.3.4.2.8)	Not applicable.	• query_acl_counter Command (Section 7.9.3.4.4.6)
Range Checkers	Allocated with the profile.	• program_acl_profile_ranges Command (Section 7.9.3.4.3.4)	• query_acl_profile_ranges Command (Section 7.9.3.4.4.2)
CDIDs and TCAM entries	See Section 7.9.3.4.2.10 and Section 7.9.3.4.2.11.	See Section 7.8.10.	See Section 7.8.10 and Section 7.9.3.4.4.7.
All except profiles CDIDs and TCAM entries	• deallocate_acl_resources Command (Section 7.9.3.4.2.9)	Not applicable.	Not applicable.

7.9.3.4.2 Resource Allocation Commands

7.9.3.4.2.1 allocate_acl_table Command (0x0C10)

The allocate_acl_table command is used by software to allocate a virtual table and its associated action entries in the ACL block.

Table 7-83. allocate_acl_table Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0C10	Command opcode.
Datalen	4-5	32	Indirect data length in bytes, with space for completion.
Return Value	6-7		Zeroed. Used for return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
TableWidth	16-17		Requested table width in bits. Minimum width is TCAP entry which is 40 bits.
TableDepth	18-19		Requested table depth.
Action Pairs for Entry	20		Amount of action pairs per table entry. The minimal valid value for this field is 1 (a single pair of actions).
TableType	21		Table type. This field is set to 0x0 for a non-concurrent table (see Section 7.9.1.4.4) allocation. For a concurrent table (see Section 7.9.1.4.3) allocation, this field specifies the amount of concurrent tables whose Alloc IDs are specified in the *Concurrent AllocIDs* field (up to 15 concurrent tables). Thus, the new allocated table is concurrent with the table IDs specified in Alloc IDs. ***Note:*** The EMP firmware blocks commands with an out-of-bound value in this field.
Reserved	22-23		Reserved.
Data Address High	24-27		Indirect data pointer high.
Data Address Low	28-31		Indirect data pointer low.

Table 7-84 specifies the format of the indirect buffer:

Table 7-84. allocate_acl_table Buffer Format

Name	Bytes.Bits	Comments
Dependent AllocIDs	0-29	Dependent tables AllocIDs. Each word in this 15-word array specifies a dependent table Alloc ID (up to 15 Alloc IDs can be specified) according to the amount specified in the *TableType* field. All unused words are set to 0xFFFF.

The EMP firmware responds with a completion command (described in Table 7-85) containing an allocation status and, if allocation succeeded, the allocated resources IDs.

Table 7-85. Completion for allocate_acl_table Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0C10	Command opcode.
Datalen	4-5	32	Indirect data length in bytes, with space for completion.
Return Value	6-7		Return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
TableWidth	16-17		Requested table width.
TableDepth	18-19		Requested table depth.
Action Pairs for Entry	20		Amount of action pairs per table entry. The minimal valid value for this field is 1 (a single pair of actions).
TableType	21		Table type. This field is set to 0x0 for a non-concurrent table (see Section 7.9.1.4.4) allocation. For a concurrent table (see Section 7.9.1.4.3) allocation, this field specifies the amount of concurrent tables whose Alloc IDs are specified in the *Concurrent AllocIDs* field (up to 15 concurrent tables). Thus, the new allocated table is concurrent with the table IDs specified in Alloc IDs. ***Note:*** The EMP firmware blocks commands with an out-of-bound value in this field.
Reserved	22-23		Reserved.
Data Address High	24-27		Indirect data pointer high.
Data Address Low	28-31		Indirect data pointer low.

Table 7-86 specify the format of the indirect buffer.

Table 7-86. Completion for allocate_acl_table Buffer Format

Name	Bytes.Bits	Comments
Alloc_ID	0-1	If *Alloc_ID* is below 0x1000, allocation failed due to unavailable resources. Otherwise, successful allocation, this field is used by firmware to identify the table allocation.
First_TCAM	2	Index of the first allocated TCAM block. This field is set to 0xFF for a failed allocation.
Last_TCAM	3	Index of the last allocated TCAM block. This index is set to the value of *First_TCAM* for a single TCAM block allocation. This field is set to 0xFF for a failed allocation.
First_Entry	4-5	Index of the first allocated entry (in both TCAMs and action memories). This field is set to 0xFFFF for a failed allocation.
Last_Entry	6-7	Index of the last allocated entry (in both TCAMs and action memories). This field is set to 0xFFFF for a failed allocation.
Action memory 0	8	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 1	9	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 2	10	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 3	11	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 4	12	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.

Table 7-86. Completion for allocate_acl_table Buffer Format [continued]

Name	Bytes.Bits	Comments
Action memory 5	13	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 6	14	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 7	15	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 8	16	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 9	17	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 10	18	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 11	19	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 12	20	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 13	21	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 14	22	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 15	23	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 16	24	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 17	25	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 18	26	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 19	27	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.

7.9.3.4.2.2 deallocate_acl_table Command (0x0C11)

The deallocate_acl_table command is used by software to release a table allocation.

Table 7-87. deallocate_acl_table Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0C11	Command opcode.
Datalen	4-5	32	Indirect data length in bytes, with space for completion.
Return Value	6-7		Zeroed. Used for return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Alloc_ID	16-17		Allocation ID of the table being released. This is the *Alloc_ID* field supplied in the completion of the allocate_acl_table command for this table.
Reserved	18-23		Reserved.
Data Address High	24-27		Indirect data pointer high.
Data Address Low	28-31		Indirect data pointer low.

The EMP firmware responds with a completion command (described in Table 7-88) containing a release status and, if deallocation succeeded, the deallocated resources IDs.

Table 7-88. Completion for deallocate_acl_table Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0C11	Command opcode.
Datalen	4-5	32	Indirect data length in bytes, with space for completion.
Return Value	6-7		Return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Alloc_ID	16-17		Allocation ID of the table being released. This is the *Alloc_ID* field supplied in the completion of the allocate_acl_table command for this table.
Reserved	18-23		Reserved.
Data Address High	24-27		Indirect data pointer high.
Data Address Low	28-31		Indirect data pointer low.

Table 7-89. Completion for deallocate_acl_table Buffer Format

Name	Bytes.Bits	Comments
Alloc_ID (result)	0-1	If *Alloc_ID* is below 0x1000, deallocation failed due to unavailable resources. Otherwise, successful deallocation.
First_TCAM	2	Index of the first released TCAM block. This field is set to 0xFF for a failed deallocation.
Last_TCAM	3	Index of the last released TCAM block. This index is set to the value of *First_TCAM* for a single TCAM block allocation. This field is set to 0xFF for a failed deallocation.
First_Entry	4-5	Index of the first deallocated entry (in both TCAMs and action memories). This field is set to 0xFFFF for a failed deallocation.
Last_Entry	6-7	Index of the last deallocated entry (in both TCAMs and action memories). This field is set to 0xFFFF for a failed deallocation.
Action memory 0	8	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 1	9	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 2	10	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 3	11	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 4	12	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 5	13	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 6	14	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 7	15	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 8	16	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 9	17	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 10	18	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 11	19	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 12	20	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 13	21	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 14	22	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 15	23	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 16	24	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 17	25	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 18	26	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 19	27	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.

7.9.3.4.2.3 allocate_acl_actionpair Command (0x0C12)

The allocate_acl_actionpair command is used by software to allocate an additional array of action pairs to an existing table allocated using the allocate_acl_table command (see Section 7.9.3.4.2.1).

Table 7-90. allocate_acl_actionpair Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0C12	Command opcode.
Datalen	4-5	32	Indirect data length in bytes, with space for completion.
Return Value	6-7		Zeroed. Used for return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Alloc_ID	16-17		Allocation ID of the table being associated with the allocated action pair. This is the *Alloc_ID* field supplied in the completion of the allocate_acl_actionpair command for that table.
Reserved	18-23		Reserved.
Data Address High	24-27		Indirect data pointer high.
Data Address Low	28-31		Indirect data pointer low.

The EMP firmware responds with the allocate_acl_actionpair completion (described in Table 7-91).

Table 7-91. Completion for allocate_acl_actionpair Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0C12	Command opcode.
Datalen	4-5	32	Indirect data length in bytes, with space for completion.
Return Value	6-7		Return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Alloc_ID	16-17		Allocation ID of the table being associated with the allocated action pair. This is the *Alloc_ID* field supplied in the completion of the allocate_acl_actionpair command for that table.
Reserved	18-23		Reserved.
Data Address High	24-27		Indirect data pointer high.
Data Address Low	28-31		Indirect data pointer low.

Table 7-92. Completion for allocate_acl_actionpair Structure Format

Name	Bytes.Bits	Comments
Alloc_ID (result)	0-1	If *Alloc_ID* is below 0x1000, allocation failed due to unavailable resources. Otherwise, successful allocation.
Reserved	2-3	Reserved.
First_Entry	4-5	Index of the first allocated entry (in both TCAMs and action memories). This field is set to 0xFFFF for a failed allocation.
Last_Entry	6-7	Index of the last allocated entry (in both TCAMs and action memories). This field is set to 0xFFFF for a failed allocation.
Action memory 0	8	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 1	9	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 2	10	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 3	11	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 4	12	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 5	13	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 6	14	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 7	15	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 8	16	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 9	17	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 10	18	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 11	19	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 12	20	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 13	21	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 14	22	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 15	23	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 16	24	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 17	25	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 18	26	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.
Action memory 19	27	Specify the order of the memory in the allocation starting from 0. If the memory is not allocated, its order is 0xFF.

7.9.3.4.2.4 deallocate_acl_actionpair Command (0x0C13)

The deallocate_acl_actionpair command is used by software to release an allocation of an action pair array associated with an existing table using the allocate_acl_actionpair command (see Section 7.9.3.4.2.3).

Note: The deallocate_acl_actionpair command cannot be used to release an action pair array allocated using the allocate_acl_table.

Table 7-93. deallocate_acl_actionpair Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0C13	Command opcode.
Datalen	4-5	32	Indirect data length in bytes, with space for completion.
Return Value	6-7		Zeroed. Used for return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Alloc_ID	16-17		Allocation ID of the table being associated with the allocated action pair. This is the *Alloc_ID* field supplied in the completion of the allocate_acl_actionpair command for that table.
Reserved	18-23		Reserved.
Data Address High	24-27		Indirect data pointer high.
Data Address Low	28-31		Indirect data pointer low.

Table 7-94. deallocate_acl_actionpair Structure Format

Name	Bytes.Bits	Comments
Reserved	0-3	Reserved.
First_Entry	4-5	Index of the first allocated entry (in both TCAMs and action memories). This field is set to 0xFFFF for a failed allocation.
Last_Entry	6-7	Index of the last allocated entry (in both TCAMs and action memories). This field is set to 0xFFFF for a failed allocation.
Action memory 0	8	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 1	9	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 2	10	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 3	11	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 4	12	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 5	13	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 6	14	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 7	15	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.

Table 7-94. deallocate_acl_actionpair Structure Format [continued]

Name	Bytes.Bits	Comments
Action memory 8	16	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 9	17	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 10	18	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 11	19	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 12	20	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 13	21	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 14	22	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 15	23	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 16	24	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 17	25	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 18	26	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 19	27	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.

The EMP firmware responds with the deallocate_acl_actionpair completion (described in Table 7-95).

Table 7-95. Completion for deallocate_acl_actionpair Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0C13	Command opcode.
Reserved	4-5		Reserved.
Return Value	6-7		Return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Alloc_ID	16-17		Allocation ID of the table being associated with the allocated action pair. This is the *Alloc_ID* field supplied in the completion of the allocate_acl_actionpair command for that table.
Reserved	18-31		Reserved.

Table 7-96. Completion for deallocate_acl_actionpair Structure Format

Name	Bytes.Bits	Comments
Alloc_ID (result)	0-1	If alloc ID is below 0x1000, deallocation failed due to unavailable resources. Otherwise, successful deallocation, deallocated *Alloc_ID*.
Reserved	2-3	Reserved.
First_Entry	4-5	Index of the first allocated entry (in both TCAMs and action memories). This field is set to 0xFFFF for a failed allocation.
Last_Entry	6-7	Index of the last allocated entry (in both TCAMs and action memories). This field is set to 0xFFFF for a failed allocation.
Action memory 0	8	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 1	9	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 2	10	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 3	11	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 4	12	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 5	13	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 6	14	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 7	15	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 8	16	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 9	17	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 10	18	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 11	19	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 12	20	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 13	21	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 14	22	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 15	23	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 16	24	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 17	25	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 18	26	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.
Action memory 19	27	Specify the order of the memory in the deallocation starting from 0. If the memory is not deallocated, its order is 0xFF.

7.9.3.4.2.5 allocate_acl_scenario Command (0x0C14)

The allocate_acl_scenario command is used by software to allocate a scenario in the ACL block and configure it. If software only wants to change a configuration of an existing scenario, it uses the update_acl_scenario command (see Section 7.9.3.4.3.1).

Due to the large data structure required for scenario configuration, the AQ command is an indirect command.

Table 7-97. allocate_acl_scenario Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0C14	Command opcode.
Datalen	4-5	84	Usable length of additional buffer (132 bytes).
Return Value	6-7		Zeroed. Used for return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-23		Reserved.
Data Address High	24-27	Buffer Address	High bits of buffer address.
Data Address Low	28-31	Buffer Address	Low bits of buffer address.

The buffer associated with the command has the format described in Table 7-98.

Table 7-98. allocate_acl_scenario Buffer Format

Name	Bytes.Bits	Comments
TCAM 0 Select0	0.0-0.6	Byte 0 selection for the TCAM key. See GL_ACL_SCENARIO_CFG_L.*SELECT0* description for more details on applicable values.
Reserved	0.7	Reserved.
TCAM 0 Select1	1.0-1.6	Byte 1 selection for the TCAM key. See GL_ACL_SCENARIO_CFG_L.*SELECT1* description for more details on applicable values.
Reserved	1.7	Reserved.
TCAM 0 Select2	2.0-2.6	Byte 2 selection for the TCAM key. See GL_ACL_SCENARIO_CFG_L.*SELECT2* description for more details on applicable values.
Reserved	2.7	Reserved.
TCAM 0 Select3	3.0-3.6	Byte 3 selection for the TCAM key. See GL_ACL_SCENARIO_CFG_L.*SELECT3* description for more details on applicable values.
Reserved	3.7	Reserved.
TCAM 0 Select4	4.0-4.4	Byte 4 selection for the TCAM key. See GL_ACL_SCENARIO_CFG_H.*SELECT4* description for more details on applicable values.
Reserved	4.5-4.7	Reserved.

Table 7-98. allocate_acl_scenario Buffer Format [continued]

Name	Bytes.Bits	Comments
TCAM 0 Chunk Mask	5	TCAM block entry masking. See GL_ACL_SCENARIO_CFG_H.*CHUNKMASK* description for more details on applicable values. This value is set to 0x0 for an unused TCAM. ***Note:*** The EMP firmware blocks attempts of a PF to enable entries/TCAM blocks which where not allocated to it using the allocate_acl_table command (see Section 7.9.3.4.2.1).
TCAM 0 StartCompare	6.0	TCAM block StartCompare setting (see Section 7.9.2.5.1 for additional details). This value is set to 0x1 for an unused TCAM.
TCAM 0 StartSet	6.1	TCAM block StartSet setting (see Section 7.9.2.5.1 for additional details). This value is set to 0x0 for an unused TCAM.
Reserved	6.2-6.7	Reserved.
TCAM 1 configuration	7-13	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 2 configuration	14-20	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 3 configuration	21-27	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 4 configuration	28-34	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 5 configuration	35-41	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 6 configuration	42-48	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 7 configuration	49-55	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 8 configuration	56-62	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 9 configuration	63-69	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 10 configuration	70-76	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 11 configuration	77-83	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 12 configuration	84-90	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 13 configuration	91-97	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 14 configuration	98-104	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 15 configuration	105-111	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
Action memory 0 TCAM association	112.0-112.6	Action memory association to a TCAM block (see Section 7.9.2.5.2 for more details) for this scenario. This field is set to 0x0 for a disabled action memory. ***Note:*** The EMP firmware blocks attempts of a PF to program an out-of-bound value in this field.

Table 7-98. allocate_acl_scenario Buffer Format [continued]

Name	Bytes.Bits	Comments
Action memory 0 enable	112.7	Action memory enable for this scenario see Section 7.9.2.5.2 for more details). ***Note:*** The EMP firmware blocks attempts of a PF to use an action memory no associated to it using the allocate_acl_table command (see Section 7.9.3.4.2.1) or the allocate_acl_actionpair command (see Section 7.9.3.4.2.3).
Action memory 1 configuration	113	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 2 configuration	114	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 3 configuration	115	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 4 configuration	116	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 5 configuration	117	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 6 configuration	118	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 7 configuration	119	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 8 configuration	120	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 9 configuration	121	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 10 configuration	122	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 11 configuration	123	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 12 configuration	124	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 13 configuration	125	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 14 configuration	126	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 15 configuration	127	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 16 configuration	128	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 17 configuration	129	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 18 configuration	130	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 19 configuration	131	Action memory configuration including the same structure defined in byte 112 for action memory 0.

The EMP firmware responds with a completion command (described in Table 7-99) containing an allocation status and, if allocation succeeded, the allocated scenario ID.

Table 7-99. Completion for allocate_acl_scenario Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0C14	Command opcode.
Reserved	4-5		Reserved.
Return Value	6-7		Return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Scenario_ID	16-17		If the value of this field is less than 0x1000, the allocation failed. Otherwise, for a successful allocation, this field holds the allocated scenario ID where the low eight bits of the ID specify the scenario index.
Reserved	18-23		Reserved.
Data Address High	24-27	Buffer Address	High bits of buffer address copied from the command.
Data Address Low	28-31	Buffer Address	Low bits of buffer address copied from the command.

7.9.3.4.2.6 deallocate_acl_scenario Command (0x0C15)

The deallocate_acl_scenario command is used by software to release a scenario allocation.

Table 7-100. deallocate_acl_scenario Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0C15	Command opcode.
Reserved	4-5		Reserved.
Return Value	6-7		Zeroed. Used for return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Scenario_ID	16-17		Scenario ID of the scenario being released. This is the *Scenario_ID* field supplied in the completion of the allocate_acl_scenario command for this scenario.
Reserved	18-31		Reserved.

The EMP firmware responds with a completion command (described in Table 7-101) containing a release status and, if deallocation succeeded, the deallocated scenario ID.

Table 7-101. Completion for deallocate_acl_scenario Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0C15	Command opcode.
Reserved	4-5		Reserved.
Return Value	6-7		Return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-31		Reserved.

7.9.3.4.2.7 allocate_acl_counters Command (0x0C16)

The allocate_acl_counters command is used by software to allocate a contiguous block of counters to be used in the actions related to its entries.

Note: After several occurrences of counter allocation and deallocation, the counters might get fragmented, which affects the ability of EMP firmware ability to allocate a large contiguous block of counters. In case software was unable to get it's initial request, it might want to repeat the allocation request with a smaller amount request.

Table 7-102. allocate_acl_counters Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0C16	Command opcode.
Reserved	4-5		Reserved.
Return Value	6-7		Zeroed. Used for return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Counter Amount	16		Amount of contiguous counters requested. ***Note:*** The minimal valid value for this field is 1. The maximal valid value is 255. If the software requires more counters, it issues an additional allocate_acl_counters command.
Counters Type	17		0x0 = Byte or Packet counters 0x1 = Byte/Packet counter duals 0x2-0xFF = Reserved
Counter Bank	18		Requested counter bank allocation. The valid values for Byte or Packet counters are 0-3. The valid values for Byte/Packet counter duals are 0-1. ***Note:*** The EMP firmware blocks attempts of a PF to program an out-of-bound value in this field.
Reserved	19-31		Reserved.

The EMP firmware responds with a completion command (described in Table 7-103) containing an allocation status and, if allocation succeeded, the allocated counter IDs.

Table 7-103. Completion for allocate_acl_counters Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0C16	Command opcode.
Reserved	4-5		Reserved.
Return Value	6-7		Return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Counter Amount	16		Amount of contiguous counters requested. ***Note:*** The minimal valid value for this field is 1. The maximal valid value is 255. If the software requires more counters, it issues an additional allocate_acl_counters command.
Counters Type	17		0x0 = Byte or Packet counters 0x1 = Byte/Packet counter duals 0x2-0xFF = Reserved
Counter Bank	18		Requested counter bank allocation. The valid values for Byte or Packet counters are 0-3. The valid values for Byte/Packet counter duals are 0-1. ***Note:*** The EMP firmware blocks attempts of a PF to program an out-of-bound value in this field.
Reserved	19		Reserved.
First_Counter	20-21		First allocated counter index. For unsuccessful allocation, this value is set to 0xFFFF.
Last_Counter	22-23		Last allocated counter index. For unsuccessful allocation, this value is set to 0xFFFF.
Reserved	24-31		Reserved.

7.9.3.4.2.8 deallocate_acl_counters Command (0x0C17)

The deallocate_acl_counters command is used by software to release a contiguous block of counters to be used in the actions related to its entries.

When using the deallocate_acl_counters command, software is not obligated to release a full counter block it allocated using the allocate_acl_counters command. It can choose to release all of it or a part of it. Since counters are a limited resource, software is encouraged to release counters it does not use.

Table 7-104. deallocate_acl_counters Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0C17	Command opcode.
Reserved	4-5		Reserved.
Return Value	6-7		Zeroed. Used for return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
First_Counter	16-17		First released counter index.

Table 7-104. deallocate_acl_counters Command [continued]

Name	Byte.Bit	Value	Remarks
Last_Counter	18-19		Last released counter index.
Counters Type	20		0x0 = Byte or Packet counters 0x1 = Byte/Packet counter duals 0x2-0xFF = Reserved
Counter Bank	21		Requested counter bank allocation. The valid values for Byte or Packet counters are 0-3. The valid values for Byte/Packet counter duals are 0-1. ***Note:*** The EMP firmware blocks attempts of a PF to program an out-of-bound value in this field.
Reserved	22-31		Reserved.

The EMP firmware responds with a completion command (described in Table 7-105).

Table 7-105. Completion for deallocate_acl_counters Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0C17	Command opcode.
Reserved	4-5		Reserved.
Return Value	6-7		Return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
First_Counter	16-17		First released counter index.
Last_Counter	18-19		Last released counter index.
Counters Type	20		0x0 = Byte or Packet counters 0x1 = Byte/Packet counter duals 0x2-0xFF = Reserved
Counter Bank	21		Requested counter bank allocation. The valid values for Byte or Packet counters are 0-3. The valid values for Byte/Packet counter duals are 0-1. ***Note:*** The EMP firmware blocks attempts of a PF to program an out-of-bound value in this field.
Reserved	22-31		Reserved.

7.9.3.4.2.9 deallocate_acl_resources Command (0x0C1A)

The deallocate_acl_resources command is used by software to release all the resources allocated for it using a single command.

Table 7-106. deallocate_acl_resources Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0C1A	Command opcode.
Reserved	4-5		Reserved.
Return Value	6-7		Zeroed. Used for return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-31		Reserved.

In response, the EMP firmware responds with a completion command (described in Table 7-107) containing a release status.

Table 7-107. Completion for deallocate_acl_resources Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0C1A	Command opcode.
Reserved	4-5		Reserved.
Return Value	6-7		Return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-31		Reserved.

7.9.3.4.2.10 Generic Allocate Resource Command

The command is fully described in Section 7.8.12.2.4, "Allocate Resource (0x0208)". This command can allocate the following resources:

- ACL profile builder CDID
- ACL profile builder TCAM table entries
- ACL profile builder Profile ID

A list of resource descriptors is supplied as a response to this admin command. For the ACL, each descriptor of two bytes specifies the number of the resource.

7.9.3.4.2.11 Generic Free Resource Command

The command is fully described in Section 7.8.12.2.5, "Free Resource (0x0209)". This command can deallocate the following resources:

- ACL profile builder CDID
- ACL profile builder TCAM table entries
- ACL profile builder Profile ID

A list of resource descriptors is supplied as a to this admin command. For the ACL, each descriptor of two bytes specifies the number of the resource.

7.9.3.4.3 Programming/Update Commands

7.9.3.4.3.1 update_acl_scenario Command (0x0C1B)

The update_acl_scenario command is used to update the context of a scenario previously allocated using the allocate_acl_scenario command (see Section 7.9.3.4.2.3).

Due to the large data structure required for scenario configuration, the AQ command is an indirect command.

Table 7-108. update_acl_scenario Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0C1B	Command opcode.
Datalen	4-5	0x84	Usable length of additional buffer (132 bytes).
Return Value	6-7		Zeroed. Used for return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Scenario_ID	16-17		Updated scenario ID. ***Note:*** The firmware blocks attempts by software to update configuration for a scenario not allocated to it.
Reserved	18-23		Reserved.
Data Address High	24-27	Buffer Address	High bits of buffer address.
Data Address Low	28-31	Buffer Address	Low bits of buffer address.

The buffer associated with the command has the format described in Table 7-109.

Table 7-109. update_acl_scenario Buffer Format

Name	Bytes.Bits	Comments
TCAM 0 Select0	0.0-0.6	Byte 0 selection for the TCAM key. See GL_ACL_SCENARIO_CFG_L.*SELECT0* description for more details on applicable values. This value is set to 0x0 for an unused TCAM.
Reserved	0.7	Reserved.

Table 7-109. update_acl_scenario Buffer Format [continued]

Name	Bytes.Bits	Comments
TCAM 0 Select1	1.0-1.6	Byte 1 selection for the TCAM key. See GL_ACL_SCENARIO_CFG_L.*SELECT1* description for more details on applicable values. This value is set to 0x0 for an unused TCAM.
Reserved	1.7	Reserved.
TCAM 0 Select2	2.0-2.6	Byte 2 selection for the TCAM key. See GL_ACL_SCENARIO_CFG_L.*SELECT2* description for more details on applicable values. This value is set to 0x0 for an unused TCAM.
Reserved	2.7	Reserved.
TCAM 0 Select3	3.0-3.6	Byte 3 selection for the TCAM key. See GL_ACL_SCENARIO_CFG_L.*SELECT3* description for more details on applicable values. This value is set to 0x0 for an unused TCAM.
Reserved	3.7	Reserved.
TCAM 0 Select4	4.0-4.4	Byte 4 selection for the TCAM key. See GL_ACL_SCENARIO_CFG_H.*SELECT4* description for more details on applicable values. This value is set to 0x0 for an unused TCAM.
Reserved	4.5-4.7	Reserved.
TCAM 0 Chunk Mask	5	TCAM block entry masking. See GL_ACL_SCENARIO_CFG_H.*CHUNKMASK* description for more details on applicable values. This value is set to 0x0 for an unused TCAM. ***Note:*** The EMP firmware blocks attempts of a PF to enable entries/TCAM blocks which where not allocated to it using the allocate_acl_table command (see Section 7.9.3.4.2.1).
TCAM 0 StartCompare	6.0	TCAM block StartCompare setting (see Section 7.9.2.5.1 for additional details). This value is set to 0x1 for an unused TCAM.
TCAM 0 StartSet	6.1	TCAM block StartSet setting (see Section 7.9.2.5.1 for additional details). This value is set to 0x0 for an unused TCAM.
Reserved	6.2-6.7	Reserved.
TCAM 1 configuration	7-13	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 2 configuration	14-20	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 3 configuration	21-27	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 4 configuration	28-34	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 5 configuration	35-41	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 6 configuration	42-48	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 7 configuration	49-55	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 8 configuration	56-62	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.

Table 7-109. update_acl_scenario Buffer Format [continued]

Name	Bytes.Bits	Comments
TCAM 9 configuration	63-69	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 10 configuration	70-76	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 11 configuration	77-83	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 12 configuration	84-90	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 13 configuration	91-97	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 14 configuration	98-104	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
TCAM 15 configuration	105-111	TCAM configuration including the same structure defined in bytes 0-6 for TCAM 0.
Action memory 0 TCAM association	112.0-112.6	Action memory association to a TCAM block (see Section 7.9.2.5.2 for more details) for this scenario. This field is set to 0x0 for a disabled action memory. ***Note:*** The EMP firmware blocks attempts of a PF to program an out-of-bound value in this field.
Action memory 0 enable	112.7	Action memory enable for this scenario (see Section 7.9.2.5.2 for more details). The EMP firmware blocks attempts of a PF to use an action memory no associated to it using the allocate_acl_table command (see Section 7.9.3.4.2.1) or the allocate_acl_actionpair command (see Section 7.9.3.4.2.3).
Action memory 1 configuration	113	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 2 configuration	114	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 3 configuration	115	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 4 configuration	116	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 5 configuration	117	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 6 configuration	118	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 7 configuration	119	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 8 configuration	120	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 9 configuration	121	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 10 configuration	122	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 11 configuration	123	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 12 configuration	124	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 13 configuration	125	Action memory configuration including the same structure defined in byte 112 for action memory 0.

Table 7-109. update_acl_scenario Buffer Format [continued]

Name	Bytes.Bits	Comments
Action memory 14 configuration	126	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 15 configuration	127	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 16 configuration	128	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 17 configuration	129	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 18 configuration	130	Action memory configuration including the same structure defined in byte 112 for action memory 0.
Action memory 19 configuration	131	Action memory configuration including the same structure defined in byte 112 for action memory 0.

The EMP firmware responds with a completion command (described in Table 7-110).

Table 7-110. Completion for update_acl_scenario Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0C1B	Command opcode.
Reserved	4-5		Reserved.
Return Value	6-7		Return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Scenario_ID	16-17		If failed value smaller than 0x1000, unsuccessful update. Otherwise, successful update, this field holds the updated scenario ID.
Reserved	18-23		Reserved.
Data Address High	24-27	Buffer Address	High bits of buffer address copied from the command.
Data Address Low	28-31	Buffer Address	Low bits of buffer address copied from the command.

7.9.3.4.3.2 program_acl_actionpair Command (0x0C1C)

The program_acl_actionpair command is used by software to program and/or update action entries.

Table 7-111. program_acl_actionpair Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0C1C	Command opcode.
Datalen	4-5	0x8	Reserved.
Return Value	6-7		Zeroed. Used for return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
action_mem_index	16		Action memory index to program/update. ***Note:*** The firmware blocks attempts by software to update an action memory not allocated to it.
Reserved	17		Reserved.
action_entry_index	18-19		The entry index in the action memory to be programmed/updated. ***Note:*** The firmware blocks attempts by software to update an entry not allocated to it.
Reserved	20-23		Reserved.
Data Address High	24-27	Buffer Address	High bits of buffer address.
Data Address Low	28-31	Buffer Address	Low bits of buffer address.

The buffer associated with the command has the format described in Table 7-112.

Table 7-112. program_acl_actionpair Buffer Format

Name	Bytes.Bits	Comments
Action0_Priority	0	Action priority. Applicable values are between 0 to 7.
Action0_MDID	1	Action metadata identifier. This field is set to 0x0 for a NOP action.
Action0_value	2-3	Action value.
Action1_Priority	4	Action priority. Applicable values are between 0 to 7.
Action1_MDID	5	Action metadata identifier. This field is set to 0x0 for a NOP action.
Action1_value	6-7	Action value.

The EMP firmware responds with a completion command described in Table 7-113.

Table 7-113. Completion for program_acl_actionpair Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0C1C	Command opcode.
Reserved	4-5		Reserved.
Return Value	6-7		Return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.

Table 7-113. Completion for program_acl_actionpair Command [continued]

Name	Byte.Bit	Value	Remarks
action_mem_index	16		Action memory index to program/update. ***Note:*** The firmware blocks attempts by software to update an action memory not allocated to it.
Reserved	17		Reserved.
action_entry_index	18-19		The entry index in the action memory to be programmed/updated. ***Note:*** The firmware blocks attempts by software to update an entry not allocated to it.
Reserved	20-23		Reserved.
Data Address High	24-27	Buffer Address	High bits of buffer address copied from the command.
Data Address Low	28-31	Buffer Address	Low bits of buffer address copied from the command.

7.9.3.4.3.3 program_acl_profile_extraction Command (0x0C1D)

The program_acl_profile_extraction command is used by software to program or update the profile-dependent configuration (see Section 7.9.2.6) for a profile allocated to it.

Due to the large data structure required for scenario configuration, the AQ command is an indirect command.

Table 7-114. program_acl_profile_extraction Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0C1D	Command opcode.
Datalen	4-5	0x4C	Usable length of additional buffer (76 bytes).
Return Value	6-7		Zeroed. Used for return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Profile ID	16		Programmed/updated Profile ID.
Reserved	17-23		Reserved.
Data Address High	24-27	Buffer Address	High bits of buffer address.
Data Address Low	28-31	Buffer Address	Low bits of buffer address.

The buffer associated with the command has the format described in Table 7-115.

Table 7-115. program_acl_profile_extraction Buffer Format

Name	Bytes.Bits	Comments
Byte selection base-select for byte 0	0.0-0.5	Byte selection for the byte selection base from the extracted fields (expressed as a byte offset in the extracted fields). Applicable values are 0-63.
Reserved	0.6-0.7	Reserved.
Byte selection base-select for byte 1	1	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 2	2	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 3	3	Byte selection using the same structure defined in byte 0 for byte 0.

Table 7-115. program_acl_profile_extraction Buffer Format [continued]

Name	Bytes.Bits	Comments
Byte selection base-select for byte 4	4	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 5	5	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 6	6	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 7	7	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 8	8	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 9	9	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 10	10	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 11	11	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 12	12	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 13	13	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 14	14	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 15	15	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 16	16	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 17	17	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 18	18	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 19	19	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 20	20	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 21	21	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 22	22	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 23	23	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 24	24	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 25	25	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 26	26	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 27	27	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 28	28	Byte selection using the same structure defined in byte 0 for byte 0.
Byte selection base-select for byte 29	29	Byte selection using the same structure defined in byte 0 for byte 0.
Word selection base-select for word 0	30.0-30.4	Word selection for the word selection base from the extracted fields (expressed as a word offset in the extracted fields). Applicable values are 0-31.
Reserved	30.5-30.7	Reserved.
Word selection base-select for word 1	31	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 2	32	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 3	33	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 4	34	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 5	35	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 6	36	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 7	37	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 8	38	Word selection using the same structure defined in byte 30 for word 0.

Table 7-115. program_acl_profile_extraction Buffer Format [continued]

Name	Bytes.Bits	Comments
Word selection base-select for word 9	39	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 10	40	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 11	41	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 12	42	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 13	43	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 14	44	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 15	45	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 16	46	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 17	47	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 18	48	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 19	49	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 20	50	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 21	51	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 22	52	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 23	53	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 24	54	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 25	55	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 26	56	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 27	57	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 28	58	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 29	59	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 30	60	Word selection using the same structure defined in byte 30 for word 0.
Word selection base-select for word 31	61	Word selection using the same structure defined in byte 30 for word 0.
Double word selection base-select for double-word 0	62.0-62.3	Double word selection for the double-word selection base from the extracted fields (expressed as a double-word offset in the extracted fields). Applicable values are 0-15.
Reserved	62.4-62.7	Reserved.
Double word selection base-select for double-word 1	63	Double word selection using the same structure defined in byte 60 for double-word 0.
Double word selection base-select for double-word 2	64	Double word selection using the same structure defined in byte 60 for double-word 0.
Double word selection base-select for double-word 3	65	Double word selection using the same structure defined in byte 60 for double-word 0.
Double word selection base-select for double-word 4	66	Double word selection using the same structure defined in byte 60 for double-word 0.
Double word selection base-select for double-word 5	67	Double word selection using the same structure defined in byte 60 for double-word 0.
Double word selection base-select for double-word 6	68	Double word selection using the same structure defined in byte 60 for double-word 0.
Double word selection base-select for double-word 7	69	Double word selection using the same structure defined in byte 60 for double-word 0.

Table 7-115. program_acl_profile_extraction Buffer Format [continued]

Name	Bytes.Bits	Comments
Double word selection base-select for double-word 8	70	Double word selection using the same structure defined in byte 60 for double-word 0.
Double word selection base-select for double-word 9	71	Double word selection using the same structure defined in byte 60 for double-word 0.
Double word selection base-select for double-word 10	72	Double word selection using the same structure defined in byte 60 for double-word 0.
Double word selection base-select for double-word 11	73	Double word selection using the same structure defined in byte 60 for double-word 0.
Double word selection base-select for double-word 12	74	Double word selection using the same structure defined in byte 60 for double-word 0.
Double word selection base-select for double-word 13	75	Double word selection using the same structure defined in byte 60 for double-word 0.
Double word selection base-select for double-word 14	76	Double word selection using the same structure defined in byte 60 for double-word 0.
PF0 scenario	77	PF0 scenario number
PF1 scenario	78	PF1 scenario number
PF2 scenario	79	PF2 scenario number
PF3 scenario	80	PF3 scenario number
PF4 scenario	81	PF4 scenario number
PF5 scenario	82	PF5 scenario number
PF6 scenario	83	PF6 scenario number
PF7 scenario	84	PF7 scenario number

The EMP firmware responds with the completion command described in Table 7-116.

Table 7-116. Completion for program_acl_profile_extraction Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0C1D	Command opcode.
Reserved	4-5		Reserved.
Return Value	6-7		Return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Profile ID	16		Programmed/updated Profile ID.
Reserved	17-23		Reserved.
Data Address High	24-27	Buffer Address	High bits of buffer address.
Data Address Low	28-31	Buffer Address	Low bits of buffer address.

7.9.3.4.3.4 program_acl_profile_ranges Command (0x0C1E)

The program_acl_profile_ranges command is used by software to program or update the range checkers profile dependent configuration (see Section 7.9.2.7) for a profile allocated to it.

Due to the large data structure required for scenario configuration, the AQ command is an indirect command.

Table 7-117. program_acl_profile_ranges Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0C1E	Command opcode.
Datalen	4-5	0x30	Usable length of additional buffer (48 bytes).
Return Value	6-7		Zeroed. Used for return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Profile ID	16		Programmed/updated Profile ID.
Reserved	17-23		Reserved.
Data Address High	24-27	Buffer Address	High bits of buffer address.
Data Address Low	28-31	Buffer Address	Low bits of buffer address.

The buffer associated with the command has the format described in Table 7-118.

Table 7-118. program_acl_profile_ranges Buffer Format

Name	Bytes.Bits	Comments
Range checker 0 low boundary	0-1	Range checker low boundary. The range checker output is negative when the value related to this range checker is lower than the low boundary.
Range checker 0 high boundary	2-3	Range checker high boundary. The range checker output is negative when the value related to this range checker is higher than the high boundary.
Range checker 0 mask	4-5	Range checker bit-mask. A value of '0' in a bit clears the relevant bit input to the range checker.
Range checker 1 configuration	6-11	Range checker configuration using the same format specified in bytes 0-5 for range checker 0.
Range checker 2 configuration	12-17	Range checker configuration using the same format specified in bytes 0-5 for range checker 0.
Range checker 3 configuration	18-23	Range checker configuration using the same format specified in bytes 0-5 for range checker 0.
Range checker 4 configuration	24-29	Range checker configuration using the same format specified in bytes 0-5 for range checker 0.
Range checker 5 configuration	30-35	Range checker configuration using the same format specified in bytes 0-5 for range checker 0.
Range checker 6 configuration	36-41	Range checker configuration using the same format specified in bytes 0-5 for range checker 0.
Range checker 7 configuration	42-47	Range checker configuration using the same format specified in bytes 0-5 for range checker 0.

The EMP firmware responds with the completion command described in Table 7-119.

Table 7-119. Completion for program_acl_profile_ranges Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0C1E	Command opcode.
Reserved	4-5		Reserved.
Return Value	6-7		Return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Profile ID	16		Programmed/updated Profile ID.
Reserved	17-23		Reserved.
Data Address High	24-27	Buffer Address	High bits of buffer address.
Data Address Low	28-31	Buffer Address	Low bits of buffer address.

7.9.3.4.3.5 program_acl_entry Command (0x0C20)

The program_acl_entry command is used by software to program or update an entry in a TCAM block allocated to it using the allocate_acl_table command (see Section 7.9.3.4.2.1).

Table 7-120. program_acl_entry Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0C20	Command opcode.
Datalen	4-5	0x8	
Return Value	6-7		Zeroed. Used for return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
TCAM Index	16		Updated TCAM block index. ***Note:*** The firmware blocks attempts by software to update a TCAM block not allocated to it.
Reserved	17		Reserved.
Entry Index	18-19		Updated entry index. ***Note:*** The firmware blocks attempts by software to update an entry not allocated to it.
Reserved	20-23		Reserved.
Data Address High	24-27	Buffer Address	High bits of buffer address.
Data Address Low	28-31	Buffer Address	Low bits of buffer address.

Table 7-121. program_acl_entry Buffer Format

Name	Bytes.Bits	Comments
Entry key	0-4	Entry key (40 bits). See Section 7.9.2.3 for more details.
Reserved	5-7	Reserved.
Entry key invert	8-12	Entry key invert (40 bits). See Section 7.9.2.3 for more details.
Reserved	13-15	Reserved.

The EMP firmware responds with the completion described in Table 7-122.

Table 7-122. Completion for program_acl_entry Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0C20	Command opcode.
Reserved	4-5		Reserved.
Return Value	6-7		Return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
TCAM Index	16		Updated TCAM block index.
Reserved	17		Reserved.
Entry Index	18-19		Updated entry index.
Reserved	20-23		Reserved.
Data Address High	24-27	Buffer Address	High bits of buffer address.
Data Address Low	28-31	Buffer Address	Low bits of buffer address.

7.9.3.4.4 Query Commands

7.9.3.4.4.1 query_acl_profile Command (0x0C21)

The query_acl_profile command is used by software to query a profile dependent configuration (see Section 7.9.2.6) for a profile allocated for it.

Due to the large data structure required for scenario configuration, the AQ command is an indirect command.

Table 7-123. query_acl_profile Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0C21	Command opcode.
Datalen	4-5	0x4C	Usable length of additional buffer (76 bytes)
Return Value	6-7		Zeroed. Used for return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Profile ID	16		Queried Profile ID.
Reserved	17-23		Reserved.
Data Address High	24-27	Buffer Address	High bits of buffer address.
Data Address Low	28-31	Buffer Address	Low bits of buffer address.

The EMP firmware responds with a completion described in Table 7-124.

Table 7-124. Completion for query_acl_profile Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0C21	Command opcode.
Reserved	4-5		Reserved.
Return Value	6-7		Return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Profile ID	16		Programmed/updated Profile ID.
Reserved	17-23		Reserved.
Data Address High	24-27	Buffer Address	High bits of buffer address.
Data Address Low	28-31	Buffer Address	Low bits of buffer address.

The buffer associated with the completion has the format described in Table 7-115.

7.9.3.4.4.2 query_acl_profile_ranges Command (0x0C22)

The query_acl_profile_ranges command is used by software to query the range checkers profile dependent configuration (see Section 7.9.2.7) for a profile allocated to it.

Due to the large data structure required for scenario configuration, the AQ command is an indirect command.

Table 7-125. query_acl_profile_ranges Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0C22	Command opcode.
Datalen	4-5	0x30	Usable length of additional buffer (48 bytes)
Return Value/VFID	6-7		Return value/VF ID for command or event. Only sent by firmware or PF driver. See Section 9.5.9.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Profile ID	16		Queried Profile ID.
Reserved	17-23		Reserved.
Data Address High	24-27	Buffer Address	High bits of buffer address.
Data Address Low	28-31	Buffer Address	Low bits of buffer address.

The buffer associated with the command has the format described in Table 7-118.

The EMP firmware responds with the completion command described in Table 7-126.

Table 7-126. Completion for query_acl_profile_ranges Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0C22	Command opcode.
Datalen	4-5		Reserved.
Return Value/VFID	6-7		Return value/VF ID for command or event. Only sent by firmware or PF driver. See Section 9.5.9.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Profile ID	16		Queried Profile ID.
Reserved	17-23		Reserved.
Data Address High	24-27	Buffer Address	High bits of buffer address.
Data Address Low	28-31	Buffer Address	Low bits of buffer address.

The buffer associated with the completion has the format described in Table 7-118.

7.9.3.4.4.3 query_acl_scenario Command (0x0C23)

The query_acl_scenario command is used by software to query the scenario dependent configuration for a scenario associated with it.

Due to the large data structure required for scenario configuration, the AQ command is an indirect command. The buffer associated with the command has the format described in Table 7-98.

Table 7-127. query_acl_scenario Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0C23	Command opcode.
Datalen	4-5	0x0	Usable length of additional buffer (132 bytes)
Return Value/VFID	6-7		Return value/VF ID for command or event. Only sent by firmware or PF driver. See Section 9.5.9.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Scenario ID	16-17		Updated scenario ID. ***Note:*** The firmware blocks attempts by software to query configuration for a scenario not allocated to it.
Reserved	18-23		Reserved.
Data Address High	24-27	Buffer Address	High bits of buffer address.
Data Address Low	28-31	Buffer Address	Low bits of buffer address.

The EMP firmware responds with a completion command (described in Table 7-128).

Table 7-128. Completion for query_acl_scenario Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0C23	Command opcode.
Datalen	4-5		Reserved.
Return Value/VFID	6-7		Return value/VF ID for command or event. Only sent by firmware or PF driver. See Section 9.5.9.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Scenario ID	16-17		
Reserved	18-23		Reserved.
Data Address High	24-27	Buffer Address	High bits of buffer address copied from the command.
Data Address Low	28-31	Buffer Address	Low bits of buffer address copied from the command.

The buffer associated with the completion has the format described in Table 7-109.

7.9.3.4.4.4 query_acl_entry Command (0x0C24)

The query_acl_entry command is used by software to query an ACL entry contents.

Table 7-129. query_acl_entry Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0C24	Command opcode.
Datalen	4-5	0x0	Reserved. 8 bytes for result.
Return Value/VFID	6-7		Return value/VF ID for command or event. Only sent by firmware or PF driver. See Section 9.5.9.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
TCAM Index	16		Updated TCAM block index. ***Note:*** The firmware blocks attempts by software to query a TCAM block not allocated to it.
Reserved	17		Reserved.
Entry Index	18-19		Updated entry index. ***Note:*** The firmware blocks attempts by software to query an entry not allocated to it.
Reserved	20-23		Reserved.
Data Address High	24-27	Buffer Address	High bits of buffer address.
Data Address Low	28-31	Buffer Address	Low bits of buffer address.

The EMP firmware responds with the completion described in Table 7-130.

Table 7-130. Completion for query_acl_entry Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0C24	Command opcode.
Datalen	4-5	0x8	
Return Value	6-7		Return Value
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
TCAM Index	16		Updated TCAM block index.
Reserved	17		Reserved.
Entry Index	18-19		Updated entry index.
Reserved	20-23		Reserved.
Data Address High	24-27	Buffer Address	High bits of buffer address.
Data Address Low	28-31	Buffer Address	Low bits of buffer address.

The buffer associated with the completion has the format described in Table 7-121.

7.9.3.4.4.5 query_acl_actionpair Command (0x0C25)

The query_acl_actionpair command is used by software to query an action pair allocated to it.

Table 7-131. query_acl_actionpair Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0C25	Command opcode.
Datalen	4-5	0x0	Reserved. 8 bytes for result.
Return Value	6-7		Zeroed. Used for return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
action_mem_index	16		Action memory index to program/update. ***Note:*** The firmware blocks attempts by software to query an action memory not allocated to it.
Reserved	17		Reserved.
action_entry_index	18-19		The entry index in the action memory to be programmed/ updated. ***Note:*** The firmware blocks attempts by software to query an entry not allocated to it.
Reserved	20-23		Reserved.
Data Address High	24-27	Buffer Address	High bits of buffer address.
Data Address Low	28-31	Buffer Address	Low bits of buffer address.

The EMP firmware responds with a completion described in Table 7-132.

Table 7-132. Completion for query_acl_actionpair Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0C25	Command opcode.
Datalen	4-5	0x8	
Return Value	6-7		Return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
action_mem_index	16		Action memory index to program/update.
Reserved	17-23		Reserved.
Data Address High	24-27	Buffer Address	High bits of buffer address.
Data Address Low	28-31	Buffer Address	Low bits of buffer address.

The buffer associated with the completion has the format described in Table 7-112.

7.9.3.4.4.6 query_acl_counter Command (0x0C27)

The query_acl_counter command is used by software to query the value of a counter allocated to it.

Table 7-133. query_acl_counter Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0C27	Command opcode.
Reserved	4-5		Reserved.
Return Value	6-7		Zeroed. Used for return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Counter Index	16-17		Queried counter index. This field is the relative counter index within the queried counter bank.
Counter Back	18		Queried counter bank.
Reserved	19-31		Reserved.

The EMP firmware responds with a completion command (described in Table 7-134).

Table 7-134. Completion for query_acl_counter Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0C27	Command opcode.
Reserved	4-5		Reserved.
Return Value	6-7		Return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Counter Index	16-17		Queried counter index. This field is the relative counter index within the queried counter bank.
Counter Back	18		Queried counter bank.
Counter Value/Packet Counter Value	19-23		This field holds the counter value.
Reserved	24-31		Reserved.

7.9.3.4.4.7 Generic Get Resource Allocation Command

The command is fully described in Section 7.8.12.2.3, "Get Resource Allocation (0x0204)". This command retrieve the allocation information for the following resources:

- ACL profile builder CDID
- ACL profile builder TCAM table entries
- ACL profile builder Profile ID

A list of resource descriptors is supplied as a response to this admin command. For the ACL, each descriptor of two bytes specify the number of the resource.

7.9.4 ACL Configuration Example

7.9.4.1 Requested Scenario

- Create a table with 64 entries.
- Create scenario to partition the table.
- create a key to access the flow entry (IPv4 5 topple).
- Create a flow entry to allow all packets with source IP 10.0.10.10 with a port range from 1000 to 50000.

7.9.4.2 Configuration Flow

7.9.4.2.1 Assumptions

GL_ACL_DEFAULT_ACT array is pre-programmed in such a way that:

- The first set of four actions are filled with non-valid actions so it does not do anything (set 0).
- The second set of four actions are filled with valid pkt_drop actions (MDID 0x8).

Profile section mechanism is pre-configured (using packages) in such a way that:

- PTYPE is transformed to PTG (PTYPE group) using XTL1 table.
  - The extractor is programmed to extract the relevant fields for every PTYPE.
  - For this example it is assumed that whenever IPv4 exists in the packets, the first two words extracted are the IPv4 source address, where its bytes are noted as SIP[0], SIP[1], SIP[2], and SIP[3].
- VSI is transformed to VSIG (VSI group) using XTL2 table.
  - Default VSIG of uninitialized VSI is 0, which causes the ACL to be by passed.

In this example, the profile that the VSI uses is noted as ACL_ProfileID.

Table 7-135 specify the configuration flow. For general table allocation see Section 7.9.3.3.2.

Table 7-135. ACL Example Command Table

Command	Input	Output	Comment
In add VSI command	Rx profile miss default action =1 Rx table miss default action =1 Tx profile miss default action =0 Tx table miss default action =0		This is set so that the default VSI behavior is to allow the Tx packet and to not allow Rx packet with table miss or profile miss.
Table Allocation			
allocate_acl_table command	TableWidth=40 (5 bytes) TableDepth=64 (entries) Action Pairs Per Entry=1 TableType =0	Alloc_ID First_TCAM Action memory 0 First Entry Last Entry	Allocate a single table of 64 entries of width five bytes (four bytes for the IP Address and one byte for range check results). The table spans over 1 TCAM. Last_Entry is First_Entry + 63. It is expected that First entry is 64 entries aligned.

Table 7-135. ACL Example Command Table [continued]

Command	Input	Output	Comment
Resource initialization			
For j=First_Entry to Last_Entry do program_acl_entry	TCAM index=First TCAM Entry index=j Entry key=0x0000000000 Entry key=0x0000000000		Make all entries of the table as never match.
for i=First_Entry to Last_Entry do program_acl_actionpair	action_mem_index=Action memory 0 action_entry_index=i Action0_Priority=0 Action1_Priority=0		Clear action memory making all action invalid.
Allocate ACL scenario			
allocate_acl_scenario	for i=0 to 15 {TCAM i Chunk Mask =0 TCAM i StartCompare=1 TCAM i StartSet=0} j=First_TCAM TCAM j Chunk Mask= 1 <<(First+Entry/64) TCAM j Startset=1 TCAM j Select0=0 (IPv4 SIP[0]) TCAM j Select1=1 (IPv4 SIP[1]) TCAM j Select2=2 (IPv4 SIP[2]) TCAM j Select3=3 (IPv4 SIP[3]) TCAM j Select4=31 (range check) for k=0 to 19 Action memory k enable=0 m=Action memory 0 Action memory m enable=1 Action memory m TCAM association=First_TCAM	Scenario_ID	Program everything into the scenario. Byte 31 looks at the range result byte.
Program/update selection base and scenario into profile			
query_acl_profile_extraction	Profile ID=ACL_ProfileID		It is assumed that the profile is already configured such that: Byte selection base-select for byte 0 =IPv4 SIP[0] Byte selection base-select for byte 1 =IPv4 SIP[1] for byte 2 =IPv4 SIP[2] for byte 3 =IPv4 SIP[3] So the first four bytes extract the source IP.
program_acl_profile_extraction	Profile ID=ACL_ProfileID i=PF number associated with the VSI PFi_scenario=Scenario_ID		Associate the scenario to the Profile for the PF of the VSI.
query_acl_profile_ranges	Profile ID=ACL_ProfileID	for i=0 to 7 { Range checker i low boundary Range checker i high boundary Range checker i mask}	According to the mask if the next available range is found. Available range is identified by mask==0. The next available range is noted by N.

Table 7-135. ACL Example Command Table [continued]

Command	Input	Output	Comment
program_acl_profile_ranges	Profile ID=ACL_ProfileID Range checker N low boundary=1000 Range checker N high boundary=50000 Range checker N mask=0xffff		Only for range checker N the parameters change.
upload section Command	Section Type Number=0x17 (XLT2 table for ACL)		Get current XLT2 table.
upload section Command	Number=0x17 (XLT2 table for ACL) Update the VSIG in XLT2 data buffer		Program VSIG for that VSI to use the profile.
Add rule to table (can be done while packets are being received)			
program_acl_actionpair	action_mem_index =Action memory 0 action_entry_index=First Entry Action0_Priority=0x7 Action0_MDID=55 (nop) Action1_Priority=0		The NOP allows the packet by doing no change. Because of the NOP, the default action is not applied.
program_acl_entry	TCAM index=First TCAM Entry index=First Entry m=1<<N; Entry key=~0x0A000A0A00|~m Entry key invert=0x0A000A0AFF		The four high bytes are pre-configured to identify 10.0.10.10. The least significant byte is programed to identify the bit set of the bit indicated by 1<<N.

7.10 Receive Classification Filters

7.10.1 Introduction

This section describes the receive classification filters that direct inbound packets to a processing engine or cluster out of the LAN engine and RDMA engine (also named as Protocol Engine or PE). The filters also define a specific queue or a context within the selected engine. All the filters described in this section relate to filtering within a specific VSI that is defined by the switch filters or ACL filters.

The classification filters and other header processing units in the device inspect the first 504 bytes of the received packets.

7.10.1.1 Association with a Packet Engine

The general rule is that a frame is tested to be PE type or LAN. Filtering to the PE is subject to a set of rules defined in the subsections of Section 7.10.9, "Protocol Engine (PE) Filters": APBVT L4 port table and PE Quad-hash filter (described in detail in Section 7.10.9.1).

7.10.1.2 Receive Classification Filters Priority and Usage

Received traffic goes through a set of filters that determine the destination of each received frame. The receive classification filters operate on frames received from the network as well as frames forwarded by the internal switch from a local port (VSI). If a frame is replicated by the internal switch, each replica goes independently through the filters.

The classification filters operate in the following priority ordering (illustrated in the Figure 7-29):

- The PE filters (described in Section 7.10.9.1) direct packets to the PE at strict highest priority.
- The ACL filters (described in Section 7.10.6), the L2 switch filters (described in Section 7.10.5) and the Flow Director filter (described in Section 7.10.8) compete on queue selection according to its programmed priority (any priority setting equals to 1 or higher). These filters can optionally define a region of queues (on the same priority of queue selection) that operate together with the hash filter.
- At lowest priority, the hash filter define a region of queues and the specific queue within the region (as described in Section 7.10.7).
- Packets that do not hit any of the above filters are posted to the default queue zero of the VSI.

Note: In nominal operation, the following filters are expected to be mutually exclusive. Meaning, the same packet is not expected to match more than one of them: PE Quad hash filter, L2 Switch filter that assigns a queue, and FD filter.

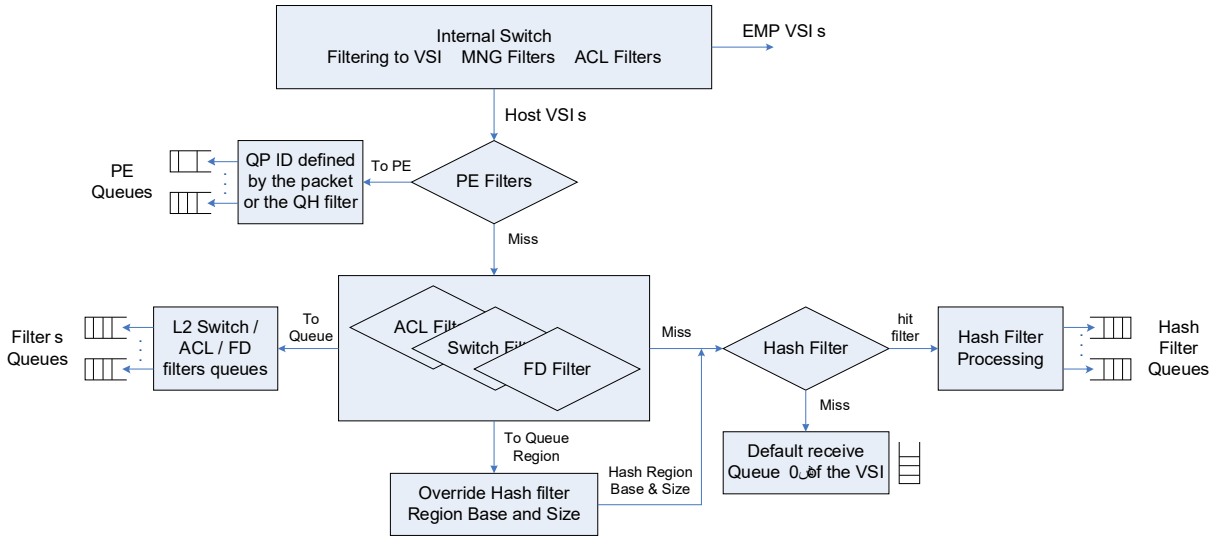


Figure 7-29. Receive Classification Filters - Top Level

Table 7-136 below shows typical use cases of the classification filters and its programming option per function.

Table 7-136. Classification Filters per PF/VF

VSI	L2 Switch Filters	ACL Filters	PE Quad Hash	Flow Director	Hash
Main Usage	VMDq1 / Control Ports / L2 Protocols	Policer	RDMA	Flex Filters[1]	Load Balance
Programming exposed to the PF	Yes	Yes	Yes[2]	Yes	Yes[3]
Programming exposed to the VF	No	No	Yes[2]	Yes[1]	No[3]

1. There are two main usages for Flow Director filters: Software driver controlled filters for Automatic Target Routing (ATR), and user/OS controlled filters. FD programming is exposed to VSIs of the PFs and the VFs as enabled by the parent PF.
2. The hardware supports RDMA offload for all PFs and up to 32 VFs (enabled by its parent PFs). Software should enable the filter only to VSIs of the PFs and the enabled VFs. Note that the software enables the VSIs, while the PE programs the specific filters.
3. Hash filters are allocated per VSI for the PFs and the VFs. Programming is enabled by the hardware for both PFs and VFs, but as a product it is exposed by the driver only for the PFs.

Table 7-137 shows the filters actions enabled per-filter type:

Table 7-137. Filter Actions by Filter Type

Action	L2 Switch Filters and ACLs	ACL Filters	PE Quad Hash	Flow Director
Target Engine: LAN, PE	LAN	PE	LAN	LAN
Receive Queue index / Context ID	Queue index within the VSI or setting a queue region for the Hash filter	QPN (QP ID)	Queue index within the VSI or setting a queue region for the Hash filter	Queue index within the region of queues within the VSI
Accept / Reject	Yes		Yes	
Increment Statistic Counter	Yes		Yes	
Flow ID	Flow ID	Report QPN (QP Index) to the PE	Report FDID in the Rx-Descriptor	Report Hash Signature in the Rx-Descriptor
Report selected bytes from the receive packet	Yes		Yes	
Flexible Actions	Yes	No	Yes	No

7.10.1.3 Resource Allocation

Some of the filter's resources can be allocated in a flexible manner between the functions, and the VSIs. allocation to the PFs are configured during run time by the Allocate Resource admin command (see Section 7.8.12.2.4).

Table 7-138 lists those parameters that are allocated using this admin command. The Flow Director table can be further allocated to VSIs by the Add VSI or Update VSI admin commands (Section 7.8.12.3.1 and Section 7.8.12.3.2, respectively).

Table 7-138. Resource ID and Types Used by the "Get Resource Allocation" Command

Resource ID	Resource Type	Allocation Type
0x20	Global RSS hash	A Global RSS LUT
0x21	Flow director Counters	Any number of counters
0x22	Guaranteed Flow director entries	Any number of filter entries
0x23	Best Effort Flow director entries	Any number of filter entries

7.10.1.4 Filter's Candidacy Rules

The Table 7-139 below list the candidacy rules for matching a filter. Complete description of the filters in the following sections.

Table 7-139. Filter's Candidacy Rules

Rule	Relevancy	Mechanism
The filter is enabled for the function	PE filters	Setting options (PFQF_PE_ENA, VPQF_PE_ENA)
The filter is enabled for the PF	FD filter	Setting options (PFQF_FD_ENA)
The filter is enabled per VSI	FD and PE filters	VSI Setting options
The filter is enabled per TC	QH filter	PF Setting options (PFQF_PE_TC_CTL)
The filter is enabled per VSI group	All filters	Packet profile selection logic

Table 7-139. Filter's Candidacy Rules [continued]

Rule	Relevancy	Mechanism
The packet's profile is enabled for the filter	All filters	Packet profile selection logic
Matched packet's VSI and packet profile	FD and QH filters	Hard -coded
Exact match of the packet's fields	FD and PE filters[1]	Matched fields are defined by the input set per packet profile.
Packet must match the APBVT	PE filters	Setting option per packet profile (GLQF_PE_CTL2)
Enable filter programming by Tx packets	FD filter	VSI Setting options
Enable filter removal by Tx FIN/RST packets	FD filter	VSI Setting option plus enabled per filter
Rx LPBK packets and Rx from the network	All filters	Both are enabled for the filters - hard-coded.

1. FD and QH filters must match complete pattern. PE filters that bypass the QH filter just match the packet profile. Hash filters that match the packet profile always pass this criteria

7.10.1.5 Filter's Candidacy Exceptions

Received packets with exception status are handled according to the rules listed below in Table 7-140.

Table 7-140. Packets Exceptions Handling[1]

Exception	Handling	Mechanism
L2 error.	Not candidate for any classification filters.	Handled by profile selection (l2_mac_error).
L3, L4 integrity error.	The packet is processed by the Hash and FD filters, but is not candidate for the PE filters.	Handled by profile selection (L3/L4 XSUM done/pass).
MPA CRC or RoCE iCRC integrity error.	Relevant only for the PE filters. Packet is posted in all cases to the PE with a hint on the CRC integrity check.	N/A
The packet includes IPv4 options or IPv6 extension headers.	Not candidate for the PE filters.	Handled by profile selection (L3/L4 XSUM not done).
The packet includes ESP extension header.	Not candidate for the PE filters.	Handled by profile selection (L3/L4 XSUM not done | pkt_is_esp). Dedicated hardware (no setting option).
The packet includes MPLS header(s).	Not candidate for the PE filters.	Handled by profile selection (outer_mpls_present_0x8847 and outer_mpls_present_0x8848).
The packet is fragmented.	Not candidate for the PE filters.	Handled by profile selection (L3/L4 XSUM not done).
Any tunneled packets.	Not candidate for the PE filters.	Handled by profile selection (L3/L4 XSUM not done).
"Drop" indication by previous units (like parser, switch, and ACL filters).	Not candidate for any classification filters.	Handled by profile selection (pkt_is_drop).
Packets indicated as DSI.	Not candidate for any classification filters.	Handled by profile selection (pkt_is_DSI).
Pre-Parser no checksum offload.	Not candidate for the PE filters.	Handled by profile selection (pprs_no_offload).
Parser Abort indication.	Not candidate for any classification filters.	Handled by profile selection (abort).
Malicious detect.	Not candidate for any classification filters.	Handled by profile selection (abort).

1. See additional candidacy criteria per filters in the following sections of the PE, Hash and FD filters

7.10.1.6 Configuration and Filter Programming Rates

Table 7-141. Performance Summary Table

Parameter	Configuration Source	Mechanism	Rate
Initial Control Domain	NVM	CSRs	Once
Profile selection parameters	NVM	CSRs	Once
Control Domain and profiles for the PE filters	NVM	CSRs	Once
Add / Remove Control Domain and profiles for the FD or RSS filters	Software via firmware	CSRs	10/Sec
PE QH Filter population	PE to hardware	Internal PE to hardware config path	Shared 0.5 M/s restriction
PE QH Filter removal	PE to hardware	Internal PE to hardware config path	
FD Filter Population	Software Application[1]	Tx packet descriptor + packet (sum of the two)	
	Software driver for ATR		
FD Filter Removal	Software[1]	Tx packet descriptor + packet	
	Auto Filter Removal by Tx or Rx traffic	Tx or Rx packet with SYN/FIN	
RSS table and Key	Software	CSR	10/sec

1. For performance considerations, FD filter population and removal by software application is expected to be made by dummy packets. Meaning, an extra 1 Mpps for population and removal.

7.10.1.7 Receive Queue Index

As indicated in the previous section, one of the filter's actions is the packet classification to LAN queues. The LAN queue indexes defined by the classification filters are within the VSI space (the VSI inherited by the switch or ACL filters). Assuming a classification filters defines a queue index 'n', then:

- VSI that is defined by the VSILAN_QTABLE (VSILAN_QBASE.*VSIQTABLE_ENA* = 1b). The queue index in the PF space equals:
  - VSILAN_QTABLE[vsi_idx, n/2].*QINDEX_0* for even 'n'
  - VSILAN_QTABLE[vsi_idx, (n-1)/2].*QINDEX_1* for odd 'n'

  Where 'n' is the VSI Queue (values in [0..15]).
- VSI that is defined by the VSILAN_QBASE (VSILAN_QBASE.*VSIQTABLE_ENA* = 0b. The queue index in the PF space equals to VSILAN_QBASE.*VSIBASE* + 'n'.
- VF queues: The mapping of the above queue indexes to relative indexes in the VF space is not a direct mapping. The mapping is defined by the queue indexes in the PF space that are programmed in the VSILAN_QTABLE and the VPLAN_QTABLE registers. For example, assuming a VF with four queues in a single VSI with the following settings:
  - VPLAN_QTABLE[0:3] = A,B,C,D (where A...D are the queue indexes in the PF space)
  - VSILAN_QTABLE[0].*QINDEX_0* = C
  - VSILAN_QTABLE[0].*QINDEX_1* = D
  - VSILAN_QTABLE[1].*QINDEX_0* = A
  - VSILAN_QTABLE[1].*QINDEX_1* = B

Then receiving a packet to queue '0' of the VSI is mapped to queue C in the PF space, which is queue index 2 in the VF space.

Note: VF queues accessed by software can be mapped as scattered queues (in the PF space) by the VPLAN_QTABLE registers or as contiguous queues (in the PF space) by the VPLAN_QBASE register.

The hash filter is a little bit more complex, as the queue index is a relative index within a specific range of queues of a VSI. See Section 7.10.7, "Hash Filter". This range of queues is also referred as "region" of queues.

Following are some exception rules for the receive queue index:

- If a frame does not match any of the classification filters below (with a queue action), the frame is directed to the default queue (queue zero of the VSI).
- The receive queue index that defines an "invalid" queue causes dropping of the packets. Such packets are counted by the GLV_RDPC counter of the VSI. The following cases are considered as "invalid" queue:
  - A VSI that is defined by VSILAN_QTABLE is limited to 16 queues (which is the size of this table). Therefore, a queue index that equals or is larger than 16 is considered "invalid".
  - The queues in a VSI that is defined by VSILAN_QTABLE are enabled by the *QINDEX_0* and *QINDEX_1* parameters. A queue index that points to a VSILAN_QTABLE entry with a value of 0x7FF is considered "invalid".
  - A VSI that is defined by VSILAN_QBASE.*VSIBASE* is limited only by the PF queue space. A queue index that exceeds the PF space is considered "invalid".

7.10.1.8 Initialization

This section lists all registers associated with the classification filters directly or indirectly and its required initialization.

Note: This section does not provide a complete description of the registers.

This description is given in the following sections as well as in the Section 13, "Programming Interface". The initialization sequence should be executed according to the following order (described in the subsections below):

1. Function Private Memory Allocation
2. Queue Allocation
3. Profiles Allocation
4. Static Classification Filters Registers
5. Dynamic Classification Filters Registers

Note: It is expected that all EMP VSIs are not enabled to any of the classification filters. Specifically the *PE_ENA* and the *FD_ENA* flags in the Add VSI commands are expected to be cleared. In addition, the EMP VSIs should be associated with VSI Groups that are not enabled for any of the classification filters.

7.10.1.8.1 Function Private Memory Allocation

The PE Quad hash filters are stored in the Function Private Memory (FPM), while part of them are fetched to the on-chip cache. The FPM allocation registers are described in Section 9.4.2, "Function Private Memory Space Configuration". These registers should be initialized per PF prior to any settings of the LAN queues and before enabling these filters.

7.10.1.8.2 Queue Allocation

LAN queues should be allocated to the functions before packets are directed to these queues. If this rule is not kept, packets directed to these queues are dropped.

PE quad hash filters, as well as TCP port filters, should be programmed before matched packets are expected. Received packets with non-matched filters are directed to the LAN queues of the function.

7.10.1.8.3 Profile Allocation

The association between packets and filters type is made through profiles. The profiles are a standard method of selecting and action based on source and type of the functions.

7.10.1.8.4 Static Classification Filters Registers

Table 7-142 describes static classification registers:

Table 7-142. Static Classification Filters Registers Initialization

CSR	Installation Comment
QH (PE) Registers	
GLQF_PEMASK_SEL	Global Classification Filter - PE Mask Select is loaded from the NVM
GLQF_PEINSET	Global Classification Filter - PE Input Set is loaded from the NVM
GLQF_PEMASK	Global Classification Filter - PE Mask is loaded from the NVM
GLQF_PE_CTL2	Global Classification Filter - PE Control 2 is loaded from the NVM
GLQF_PE_CTL	Global Classification Filter Control
GLQF_PE_OSR_STS	Global PE Classification Filter outstanding request counter
GLQF_APBVT	Global Classification Filter Accelerated Port Bit Vector
GLQF_PE_OSR_LIM	QH FLU - PCIe/FOC Outstanding Request limits
VSIQF_PE_CTL1	VSI Classification Filter - PE Control 1 (programmed by add VSI)
PFQF_PE_CTL2	PF Classification Filter - PE Control
PFQF_PE_TC_CTL	PF Classification Filter - QH TC Enable
PFQF_PE_CTL1	PF Classification Filter - PE Control
PFQF_PE_ST_CTL	PF Control register for the Statistic counter
PFQF_PE_ENA	PF Classification Filter - PE Enable
VPQF_PE_ENA	VF Classification Filter - PE Enable
VPQF_PE_CTL2	PF Classification Filter - PE Control
VPQF_PE_FLHD	VF Free List head array
VPQF_PE_CTL1	VF Classification Filter - PE Control

Table 7-142. Static Classification Filters Registers Initialization [continued]

CSR	Installation Comment
FD Registers	
GLQF_FDEVICTENA	Global Classification Filter - FD Profile Evict Enable is loaded from the NVM
GLQF_FDSWAP	Global Classification Filter - FD SWAP is loaded from the NVM
GLQF_FDMASK_SEL	Global Classification Filter - FD Mask Select is loaded from the NVM
GLQF_FDMASK	Global Classification Filter - FD Mask is loaded from the NVM
GLQF_FDINSET	Global Classification Filter - FD Input Set is loaded from the NVM
GLQF_FD_SIZE	Global Classification Filter - FD space Size is loaded from the NVM
GLQF_FDEVICTFLAG	Global Classification Filter - FD Flag Evict Enable is loaded from the NVM
PFQF_FD_SIZE	PF Classification Filter - FD space Sizes is loaded from the NVM
PFQF_FD_ENA	PF Classification Filter - FD Enable: programmed by Add VSI
VSIQF_FD_SIZE	VSI Classification Filter - FD VSI space Sizes is loaded from the NVM
VSIQF_FD_DFLT	VSI Classification Filter - FD Default Action: programmed by Add VSI command.
VSIQF_FD_CTL1	VSI Classification Filter - FD Control 1: programmed by Add VSI command.
Hash Registers	
GLQF_HSYMM	Global Classification Filter - Symmetric Hash is loaded from the NVM
GLQF_HMASK_SEL	Global Classification Filter - Hash Mask Select is loaded from the NVM
GLQF_HMASK	Global Classification Filter - Hash Mask is loaded from the NVM
GLQF_HINSET	Global Classification Filter - Hash Input Set is loaded from the NVM
GLQF_HLUT_SIZE	Global Classification Filter - Hash LUT Size
GLQF_PROF2TC	Global Classification Filter - Packet Profile to Hash TC Region Mapping
PFQF_HLUT_SIZE	PF Classification Filter - Hash LUT Size
VSIQF_HASH_CTL	VSI Classification Filter - Hash Control (programmed by add VSI)
VSIQF_TC_REGION	VSI Classification Filter - Receive TC Queue Regions: programmed by Add VSI command

7.10.1.8.5 Dynamic Classification Filters Registers

Table 7-143 lists all of the dynamic classification table registers.

Table 7-143. Dynamic Classification Filters Registers Initialization

CSR	Installation Comment
QH (PE) Registers	
GLQF_PEPIPE_CLR	Global Classification Filter - PE Pipe Clear
GLQF_PETABLE_CLR	Global Classification Filter - PE Table Clear
FD Registers	
GLQF_FDPIPE_CLR	Global Classification Filter - FD Pipe Clear
GLQF_FDTABLE_CLR	Global Classification Filter - FD Table Clear
Hash Registers	
GLQF_HLUT	Global Classification Filter - Hash LUT
GLQF_HKEY	Global Classification Filter - Hash Key

Table 7-143. Dynamic Classification Filters Registers Initialization [continued]

CSR	Installation Comment
PFQF_HLUT	PF Classification Filter - Hash LUT: programmed by Set RSS LUT command
VSIQF_HLUT	VSI Classification Filter - Hash LUT: programmed by Set RSS LUT command.
VSIQF_HKEY	VSI Classification Filter - Hash Key: programmed by Set RSS LUT command

7.10.2 Block Diagram

A top-level block diagram of the classification filters is illustrated in Figure 7-30 and described below.

Each of the filters: FD filter, QH filter, and Hash filter has its own dedicated logic.

- **Profile Chooser** and **Field Extractor** — The building blocks of the Fields Extractor logic are similar to those ones used by the switch filters and the ACL filters. The extractor logic extracts a local Field Vector (FV) from the packet headers based on the packet profile (Profile ID) per filter. The Profile ID is a function of the packet structure (packet type and flags) that is configured per filter per VSI group and per control domain that are defined per VSI.
- **Field Vector (FV) and Input Set** — Input set is a set of fields from the packet that are used for the packet classification. The input set is taken from the FV while some of the fields might be partially masked and some of the fields might be swapped or XORed. See details in Section 7.10.4.
- **Filter Lookup** — In this stage the lookup is performed either by an exact match or hash distribution (RSS). This associates the packet with a set of actions.
- **Filter Action** and **Priority Resolver** — Filter action can be classification to engine (LAN/PE), selecting a target queue, pass/reject the packet and other actions listed in Table 7-137. The action is defined by the filter with highest priority defined by the priority resolver. The default setting of the priority resolver is illustrated in Figure 7-29.

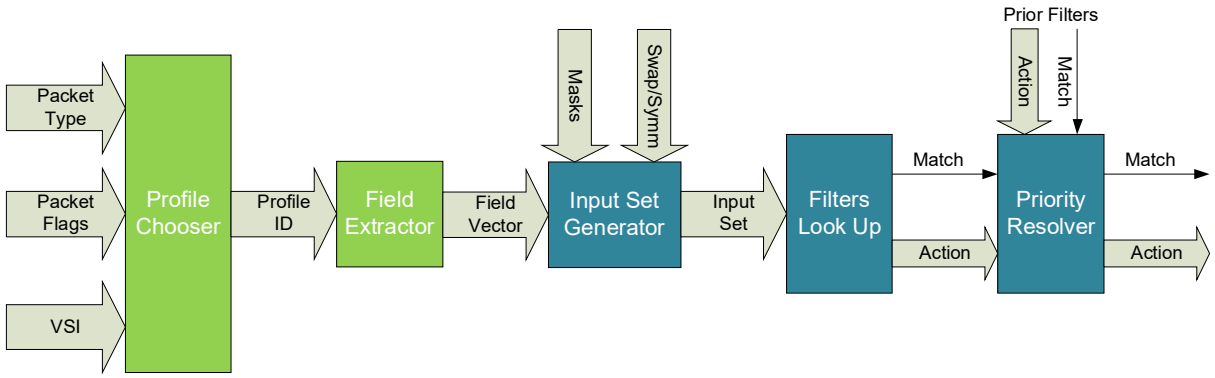


Figure 7-30. Classification Filters Block Diagram

7.10.3 Profile Chooser

- **Control Domain (CDID)** and **VSI Group (VSIG)** — The VSIs are mapped to control domains and VSI groups that are used by the extractor logic. Then the CDIS and VSIG maps the Packet Type Groups (PTGs) and the packet flags (FLGs) to packet profiles.
- **Packet Types** and **Flags** — Some combinations of the packet headers are defined by unique identification called "packet types". A portion of these packet types are defined by a smaller set of PTGs. The PTG tables map the packet types per filter per, per control domain and per VSI to a small set of 64 PTGs. On top of the PTG, there is some portion of the FLGs that are used for the classification task. A set of 16 FLGs are selected from the global set of flags per filter and per control domain.

Figure 7-31 shows the default profile chooser diagram.

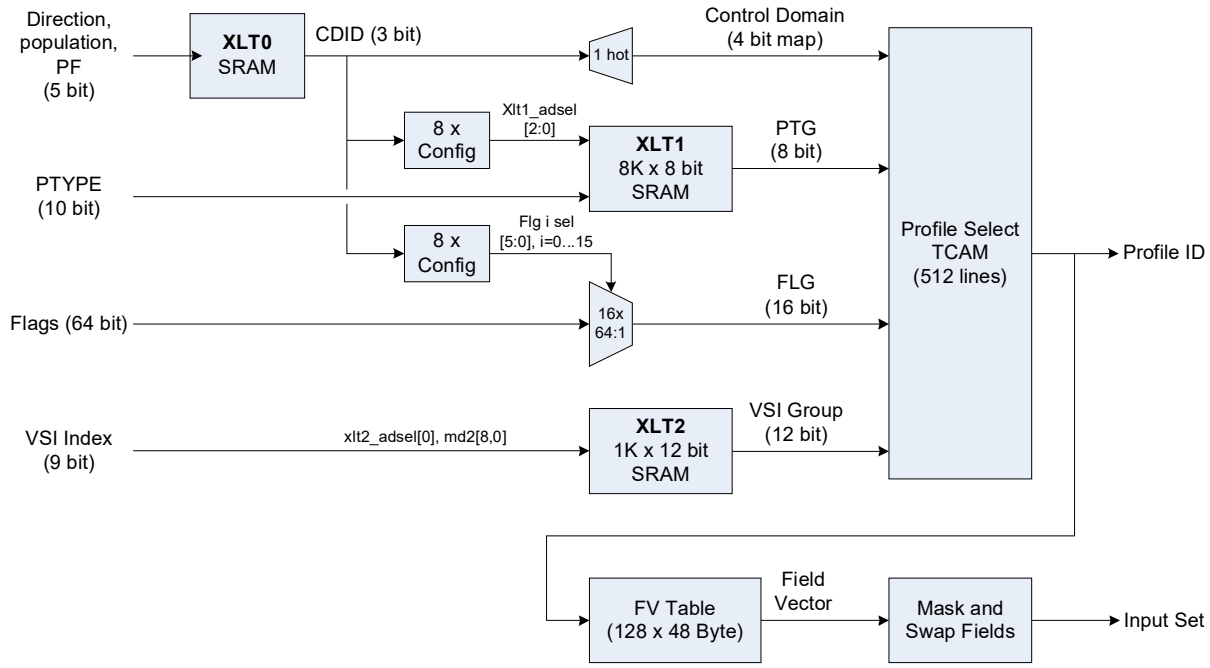


Figure 7-31. Profile Chooser - Packet Types to Profile ID and Input Set Flow

7.10.3.1 Settings for the Classification Filters

See default setting of the packet Profile IDs and the field vectors for the FD, Hash, and PE in Section 7.10.13, "Default Extractor Configuration".

7.10.4 Input Sets Generator

The input set is cleared to zero's for every packet (before the fields are posted to it from the field vector). The hardware supports any mapping between the field vector to the input set. Yet, in most cases (default setting), the input set equals exactly to the field vector. Cases in which the input set differs from the field vectors relates to swapped fields, masked fields, and asymmetric input sets. Bit fields that are masked, are replaced by zeros in the input set.

7.10.4.1 Generating PE Input Set from the Field Vector

Unique parameters of the FV to input set extraction of the PE filter are:

- The FV is mapped to the Input Set vector by the GLQF_PEINSET registers. Default settings of these registers (loaded from the NVM) map all valid words in the FV to the Input Set vector at the same word offsets. The GLQF_PEINSET is a two-dimensional array: 32 profiles x 24 words offsets per profile.
- Masking option. The mask registers are used to mask bit fields per words in the FV. There is a global set of 16 x mask registers for the PE filter. A mask register can mask out any selected bits in a FV word. Multiple mask registers can be selected for a packet profile. Multiple profiles can share the same registers. The mask options are defined by the following registers:
  - 16 x GLQF_PEMASK registers — Each register defines the byte index in the input set that should be masked and the 16-bit masking.
  - 32 x GLQF_PEMASK_SEL registers — Register 'n' in the array is associated with packet profile 'n'. It defines the GLQF_PEMASK registers that should be used for profile 'n'.

  Note: By default setting (loaded from the NVM), none of the words in the input set are masked.

7.10.4.2 Generating FD Input Set from the Field Vector

Unique parameters of the FV to input set extraction of the FD filter are:

- The FV is mapped to the Input Set vector by the GLQF_FDINSET registers. Default settings of these registers (loaded from the NVM) map all valid words in the FV to the Input Set vector at the same word offsets. The GLQF_FDINSET is a two-dimensional array: 128 profiles x 24 words offsets per profile.
- The swap option is defined by the GLQF_FDSWAP registers. The swap functionality is meaningful only at FD programming time with the GLQF_FDINSET registers. If the *SWAP* flag in the programming descriptor is set, the input set is defined by the GLQF_FDSWAP registers. Otherwise, it is defined by the GLQF_FDINSET registers.
- Masking option. The mask registers are used to mask bit fields per words in the input set. There is a global set of 32 x mask registers for the FD filter. A mask register can mask out any selected bits in an input set word. Multiple mask registers can be selected for a packet profile. Multiple profiles can share the same registers. The mask options are defined by the following registers:
  - 32 x GLQF_FDMASK registers — Each register defines the byte index in the input set that should be masked and the 16-bit masking.
  - 128 x GLQF_FDMASK_SEL — Register 'n' in the array is associated with packet profile 'n'. It defines the GLQF_FDMASK registers that should be used for profile 'n'.

  Note: By default setting (loaded from the NVM), none of the words in the input set are masked.

7.10.4.3 Generating Hash Input Set from the Field Vector

Unique parameters of the FV to input set extraction of the Hash filter are:

- The FV is mapped to the Input Set vector by the GLQF_HINSET registers. Default settings of these registers (loaded from the NVM) map all valid words in the FV to the Input Set vector at the same word offsets. The GLQF_HINSET is a two-dimensional array: 128 profiles x 24 words offsets per profile.

- Symmetric hash is defined by the GLQF_HSYMM registers. Symmetric hash is enabled for a VSI (by the Hash Scheme field in the Add VSI AQC). The symmetric hash registers structure is identical to the GLQF_HINSET registers with one difference: the *Valid* flag in the GLQF_HINSET registers is replaced by a *SYMM_ENA* flag in the GLQF_HSYMM registers. For each word with active *SYMM_ENA* flag, the input set is defined by a XOR function of the words defined by the matched GLQF_HSYMM and GLQF_HINSET registers.
- Masking option. The mask registers are used to mask bit fields per words in the input set. There is a global set of 32 x mask registers for the Hash filter. A mask register can mask out any selected bits in an input set word. The masking operation is executed after any possible symmetric action described above. Multiple mask registers can be selected for a packet profile. Multiple profiles can share the same registers. The mask options are defined by the following registers:
- 32 x GLQF_HMASK registers — Each register defines the byte index in the input set that should be masked and the 16-bit masking.
- 128 x GLQF_HMASK_SEL — Register 'n' in the array is associated with packet profile 'n'. It defines the GLQF_HMASK registers that should be used for profile 'n'.

Note: By default setting (loaded from the NVM), none of the words in the input set are masked.

7.10.5 Switch Filters

The switch filters are enabled only for packets that are not rejected for filtering by exception rules that are listed in Section 7.10.1.5. For queue-related programming in the switch, see Section 7.8.12.3.1.

7.10.6 ACL Filters

The ACL filters are enabled only for packets that are not rejected for filtering by exception rules that are listed in Section 7.10.1.5.

Some of the ACL filters can be programmed to define a target queue or a target region of queues for the hash filter. Section 7.9.3.4.3.2 for details.

7.10.7 Hash Filter

The hash filter is a mechanism to statistically distribute received packets into several receive queues. Software allocates the queues among the different processors, therefore sharing the load of packet processing among several processors. One of the most common use cases of hash filters is the RSS hash defined by Microsoft and used widely by other operating systems as well. Other generic hash functions are used by embedded systems as well. The hash filter directs the received packets to queue index within a "region" of queues of the VSI, as illustrated in the Figure 7-32. Listed below are some terms relating the hash filter.

Hash Filter Candidate Rules

The receive packet is not rejected for the filter by the exception rules listed in Section 7.10.1.5. The Hash filter is enabled per VSI group as shown in Section 7.10.13.1 (the VSI is associated to a VSI group (VSIG) that is enabled for the Hash filter by profile selection mechanism).

The hash filter priority selecting a queue is hard-coded as the lowest priority between ACL filter, PE filter, L2 Switch filter, and FD filter.

Queue Index

The queue index within the VSI is defined by the sum of the Region base and the output of the hash lookup table (modulo the region size).

Queue Region

A region of queues is defined by two parameters: Region base and Region Size. The region of queues can be defined by one of the following options in priority order:

1. **Switch filters** with *ToQueue* parameters greater than zero. In this case the region base is the "Queue Number" parameter and the region size is equal to 2^(ToQueue). Priority is defined by the *ToQueue_PRIO* of the switch filter. If the packet does not match the RSS filter, the packet is directed to queue zero of the VSI.
2. **ACL filters** with *ToQueue* parameters greater than zero - same functionality as the switch filter. In this case, the region base is the "Queue Number" parameter and the region size is equal to 2^(ToQueue), (same as the switch filters). Priority is defined by the *ToQueue_PRIO* of the FD filter.
3. **FD filter** with *ToQueue* parameter greater than zero - same functionality as the switch filter. In this case, the region base is the "Queue Number" parameter and the region size is equal to 2^(ToQueue), (same as the switch filters). Priority is defined by the *ToQueue_PRIO* of the FD filter.
4. The **packet's profile** is indicated as "Override TC" by one of 32 global GLQF_PROF2TC tables. These tables are enabled for VSIs by the TC override option and *Profile Override TC* fields in the Add VSI admin command. In this case, the region base and size are defined by the TC region described below. Priority is lower than the above filters switch and FD filters, and higher than the TC of the packet.
5. The **TC of the packet** defines the queue region. The region base and size are defined by the number and offset of Rx-Queues per TCs parameters in the Add VSI admin command. Priority is lowest of all the above options.

Hash LUT

The queue index in the region is defined by a hash value on the input set of the packet. The LS bits of the hash value selects an entry in the hash LUT. Then the LS bits of the programmed value in the LUT is the queue index within the region. The number of bits used to select the entry in the LUT match the LUT size. The number of LS bits taken from the LUT are defined by the region size. The E810 supports three sizes of Hash LUTs. Any of these LUT options is selected per VSI according to the RSS LUT selection field in the Add VSI command.

- 8 LUTs (one table per PF) that support 256 queues per region. These tables are 2K entries and support a reduces size of 512 entries and a reduced size of 128 entries as well (controlled by the *HSIZE* field in the PFQF_HLUT_SIZE registers).
- 16 global LUTs of 512 entries that support 64 queues per region. These tables support a reduces size of 128 entries (controlled by the *HSIZE* field in the matched GLQF_HLUT_SIZE registers). Selecting the Global LUTs, a specific table is selected by the RSS global LUT parameter in the Add VSI command.
- Per VSI LUT with 64 entries that support 16 queues per region.

Hash Key

See Section 7.10.10.1, "Microsoft Toeplitz-Based Hash".

Hash Function

The hash is calculated on the input set words. Masked bits within the input set are replaced by zeros for the hash calculation (as indicated in Section 7.10.4). The Hash signature is used to select a queue index as explained above. It can also be reported in the receive descriptor as detailed in Section 7.6.3.

The E810 supports one of the following 32-bit hash schemes selected per-VSI by the *Hash Scheme* field in the Add VSI command

- **Toeplitz Hash** — The hash key for the VSI is defined by the VSIQF_HKEY registers (detailed in Section 7.10.10).
- Symmetric Toeplitz Hash — The hash key is defined as above by the VSIQF_HKEY registers. Symmetric hash provides the same signature for packets in both Tx and Rx direction. It is done by a XOR function between "source" and "destination" tuples (see description of the GLQF_HSYMM registers in Section 7.10.4.3 and Section 7.10.10).
- **XOR Hash** — Simple XOR function between the DWords of the input set. Furthermore, 16-bit XOR function is made if the required XOR output is 16 bits or less and further 8-bit XOR is made if the required XOR output is 8 bits or less.
- **Jenkin's Hash** — This options is provided only for small input sets up to four bytes long (detailed in Section 7.10.10).

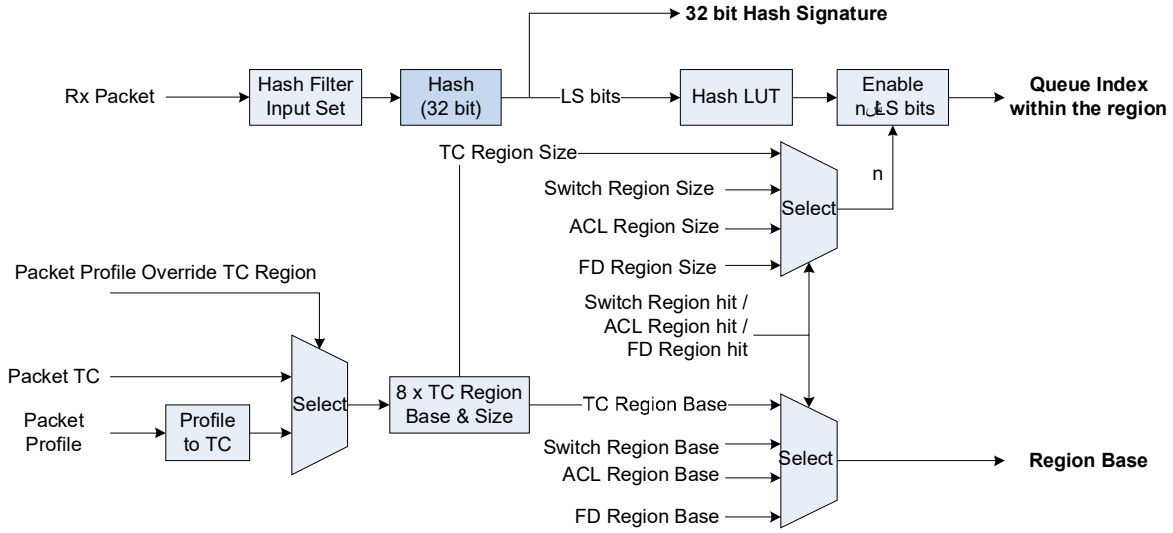


Figure 7-32. Hash Filter Block Diagram

7.10.8 Flow Director (FD) Filter

The Flow Director (FD) filter is aimed to match specific flow or flows with "some" action. The FD filter is based on "exact" match of the selected tuples named as "input set" for filtering purposes. The FD filter is composed of the following components.

FD Filter Candidacy Rules

The FD filtering is enabled by the following parameters:

- The receive packet is not rejected for the filter by the exception rules listed in Section 7.10.1.5.

- The FD filter is enabled for the parent PF by the *FD_ENA* flag in the PFQF_FD_ENA register.
- The FD filter is enabled for the VSI by the *FD_ENA* flag in the Add VSI admin command.
- The FD filter is enabled per VSI group as shown in Section 7.10.13.1 (the VSI is associated to a VSI group that is enabled for the FD filter by the *VSIG_FD* parameter in the Add VSI admin command).
- Filtering is also qualified per packet for matched profile before a search in FD table is triggered.

FD Filter Match Criteria

Packet matches a filter entry if the following packet's parameters match the filter context:

- Input Set fields.
- Packet's VSI and Packet profile.

Filter Action

The filter actions are defined by specific fields in the FD programing descriptor and are listed here (see Section 10.5.3.3 for FD programming descriptor structure):

- Pass/Reject the packet is defined by the *DROP* flag.
- Destination queue or region of queues for the hash filter and its priority are defined by the *QINDEX*, *ToQueue* and *ToQueue_PRIO* parameters in the FD programming descriptor. See Figure 7-32 for details on region of queues action.
- Statistic counters (see Section 7.10.8.1 for more details). Packet and byte counters are enabled by the *STAT_CNT* and *STAT_ENA*. The *STAT_ENA* field could be set to one of the following: No stat counter:
  - Packet counter
  - Both packet and byte counters
- Report the FD filter ID (FDID) and its priority in the Rx-Descriptor are defined by the *FDID_VAL*, *FDID_MDID*, and *FDID_PRIO* parameters in the FD programming descriptor.
- Post selected fields from the packet or metadata of the packet to the Rx-Descriptor. The selected fields that should be posted are defined by the *DESC_PROF* and *DESC_PROF_PRIO* parameters in the FD programming descriptor. For more details on the Rx-Descriptor profiles, see Section 7.6.4.

Default Filter Action

Packets that are subjected to look up in the FD table but do no match any entry are handled according the default FD action.

Note: In case of any default action other than drop, the relevant Rx-Descriptor reflects no hit occurred in the FD table.

The default is defined by the following parameters in the Add VSI admin command:

- Pass/Reject the packet is defined by the *Default Drop* flag.
- Destination queue or region of queues parameters with a priority option as defined for packets that hit the FD table:
  - Default Queue
  - Default queue group
  - Default Priority.

- If the *Default Drop* flag is cleared and the *Default Priority* equals to zero, packets that miss the FD table are handled by the next classification filter(s) according to its priority order, as shown in Figure 7-29.

FD Programming Enable

FD filters programming is enabled by the following parameters:

- Programming is enabled for the parent PF by the *FD_ENA* flag in the PFQF_FD_ENA register.
- Programming is further enabled for the programming VSI by the Flow Director programming enable flag in the Add VSI command.
- Programming is further enabled for the VSI defined by the *FD_VSI* field in the programming descriptor for the best effort space or guaranteed space in the table by the *Max Dedicated* and *Max Shared* parameters in the Add VSI command. Best effort and guaranteed spaces are described in this section.
- All conditions for the packet match the FD filter criteria including enable for the VSI group, matched profile, and no errors.
- Programming request that failed the above conditions is ignored silently.

Programming

FD filter programming is done through packets that pass through the FD. The packet causes an FD Filter programming when it is associated with a "Flow Director Programming Descriptor" described in Section 10.5.3.3 followed by packet structure that contains the filter fields.

- The descriptors used for programming (in order) are:
  - Optional Context Descriptor
  - Optional IPsec context descriptor
  - FD filter programming descriptor
  - Tx Data descriptor

  Note: Programming of the FD table cannot occur if the programming packet (described above), is dropped by any of the pipeline mechanisms.
- Programming can be selected to be on the Guaranteed space or Best Effort space according to the *FD_SPACE* field in the FD programming descriptor. It can be one of the following options:
  - Guaranteed
  - Best Effort
  - Prefer-Guaranteed
  - Prefer-Best Effort
- Programming completion status can be reported back (by the "Programming Status" Descriptor described in Section 10.4.2.2.3). It is enabled by the *COMP_REP* field in the programming descriptor that can be one of the following options:
  - No completion status
  - Completion status on failed programing
  - Completion status at all cases.

- The receive queue for the completion status is selected by the *COMP_QUEUE* flag in the FD programming descriptor. It selects between queue zero of the programming VSI or *Reporting Queue* field in the Add VSI admin command.
- *ToQueue* field in the FD programming descriptor. If there are no descriptors in the target receive queue the completion indication is dropped and counted as a dropped packet by the GLV_RDPC counter of the VSI.
- The packet that is used to program the filter can be a single packet or a TSO and it can be transmitted or not depending on the *Dummy* flag in the transmit data descriptor.
- Reprogramming an existing filter updates its parameters.
- The "source" and "destination" fields in the transmitted packets of the filter programming are usually presented in a reversed order with respect to the expected received packets. It can be adjusted by the device.

Filter Removal

Filters are removed in one of the following options:

- **Remove a single filter** by the software using the FD filter programming sequence (as above).
- **Auto-evict a single filter** by the hardware following a transmission or reception of TCP packet with *FIN* or *RST* flags. Packets that are candidates for auto-eviction are enabled per-profile by the GLQF_FDEVICTENA. The TCP flags that matters for auto-eviction are defined by the GLQF_FDEVICTFLAG register. Filters are enabled for auto-evict by the *EVICT_ENA* flag in the filter programming descriptor. The auto-eviction is also qualified per VSI by the appropriate VSIQF_FD_CTL1.*EVICT_ENA* bit and per profile by the appropriate GLQF_FDEVICTENA.
- **PF Reset** — Auto-evict all filters of the PF following a PF reset.
- **VF Reset** — As part of the VF reset flow the parent PF software is expected to initiate a request to remove all the filters of all the VSIs of the VF.
- **VM Reset** — As part of the VM reset flow the parent PF software is expected to initiate a request to remove all the filters for the VM VSI.
- **Software initiated request to remove all filters of the PF(VSI of the PF) or a VSI** — It is done by initiating the Clear FD Table admin command. Removing the filters of a PF removes the filters of all VSIs of the PF. This action can be done during nominal operation without impacting the functionality of other entities.

VSI Rules - Summary

Table 7-144 summarizes the VSI's FD actions and validity rules:

Table 7-144. VSI Rules - Summary

Action/Validity Check	The VSI by which the Action/Validity is Checked / Configuration Completion
Enable for programming	*CFG_ENA* for the source VSI and FD_ENA for the parent PF.
Programming Budget	Guaranteed and Best Effort spaces for the VSI defined by the *FD_VSI* field in the FD programming descriptor.
Enable for Filtering	*FLT_ENA* for the destination VSI + *FD_ENA* for the parent PF.
VSI in the FD filter context	Equals to the *FD_VSI* field in the FD programming descriptor.
Tx ATR filter removal	The source VSI from which the packet is sent is compared against the VSI in the filter context.
Rx ATR filter removal	The destination VSI is compared against the VSI in the filter context.
Configuration Completion	Completion is reported to the same VSI that initiated the configuration/removal command.

FD Table Size and Space

The FD table is stored in the device and can hold up to 16K filters. There are two spaces in the FD table: Guaranteed space and Best Effort space.

- **Global Setting** — These spaces are defined for the whole device by the GLQF_FD_SIZE.*FD_GSIZE* and GLQF_FD_SIZE.*FD_BSIZE* registers. Each space can be defined up to 16K filters as long as the sum of the two does not exceed the table size (of 16K filters).
- **Per PF Setting** — Each PF can be allocated a guaranteed space and best effort space by the PFQF_FD_SIZE.*FD_GSIZE* and PFQF_FD_SIZE.*FD_BSIZE* registers. Each space can be defined up to 16K filters as long as the sum of the Guaranteed spaces of all the PFs do not exceed the global guaranteed space and as long as the best effort of each PF does not exceed the global best effort space.
- **Per VSI Setting** — Each VSI can be allocated a guaranteed space and best effort space by the VSIQF_FD_SIZE.*FD_GSIZE* and VSIQF_FD_SIZE.*FD_BSIZE* registers. Each space can be defined up to 16K minus 1 filters as long as the sum of the Guaranteed spaces of all the VSIs of a PF do not exceed guaranteed space of its parent PF, and as long as the best effort of each VSI does not exceed its parent PF's best effort space.

At programming, the software can chose the space on which the filter is counted. If there is no space on the selected space, the programming request is rejected.

The software can track the number of programmed filters.

- **Per PF** — By PFQF_FD_CNT.FD_GCNT and PFQF_FD_CNT.FD_BCNT counters.
- **Per VSI** — By VSIQF_FD_CNT.FD_GCNT and VSIQF_FD_CNT.FD_BCNT counters.
- **Globally** — By the GLQF_FD_CNT.FD_GCNT and GLQF_FD_CNT.FD_BCNT counters.

FD Filter Context

Each filter entry is consisted of 64 bytes with the following parameters:

- **Parameters extracted from the packet** — Input Set (48 bytes) and the Matched packet profile.
- **Parameters from the programming descriptor** — Target Queue and control, Filter ID and control; Packet's VSI, Statistic Counter, Receive Descriptor Profile and control, and Other control flags.
- **Self table management parameters** — Linked list pointers.
- **Self storage managed parameters** — PF/VF/VSI identification, *Valid* flag, and *Dirty* flag.

FD Filter Match Criteria

Packet matches a filter entry if the following packet's parameters match the filter context:

- Input Set fields
- Packet's VSI
- Packet profile

Hash and Buckets

The FD table is organized in buckets. Buckets are sets of filters addressed by a hash function on the input set (Toeplitz hash using a global key defined by the GLQF_HKEY registers). Multiple filters might share the same hash. In such case the hardware checks for perfect match between these filters one entry at a time. The number of searches is compared against the FDLONG parameter in the GLQF_FD_CTL register.

7.10.8.1 Statistic Counters

There are 8K statistic counters that can be used by the FD filters. These counters are organized in two banks of 4K counters each. The statistic counters are allocated in blocks of 256 counters to the PFs by the Allocate Resource admin command (resource type = Flow Director counter block). An FD filter can assign no counters; a single counter (counting bytes of packets) or count both (bytes and packets). Assigning a single counter, software can select any counter out of the two banks. Assigning two counters, software defines a single counter index in the two banks for these counters. In this case, the packets are counted in bank zero and the bytes are counted in bank one.

7.10.8.2 Performance

7.10.8.2.1 Programming and Removal Performance Targets

The FD filter is expected to be used by two main usage models. Filters programming initiated by software application (above the driver). And filters programmed and owned by the driver (usually for ATR functionality).

Expected filter programming rate initiated by software application is in the range of up to 0.5M filters per second. Same rate of removing filters. Targeting Si performance for programming and removing filters at double rate of the above.

Expected filter programming by the driver is in the order of once every 20 packets. Expected auto-removal by FIN/RST is in the order of 1M filters per second.

Notes: The hardware does not provision the rate of programming nor the rate of transmit FIN/RST packets that are candidates for auto-removal of filters from the table. It is the responsibility of software to follow the above programming and auto-removal performance. Exceeding these rates, might impact transmission and reception rates. Therefore, the PF driver should enable direct programming and auto filter removal only for trusted VSIs. The PF driver could add and remove FD filters for non-trusted VSIs that are not enabled for direct programming (non-trusted VFs).

The interface by which these VFs request FD filter programming and remove FD filters is outside the scope of this document.

7.10.9 Protocol Engine (PE) Filters

The PE filters direct received packets to the Protocol Engine. It is mainly (but not limited) to iWARP and RoCEv2 traffic. The PE filter is composed of an APBVT filter and a Quad Hash (QH) filter described below.

Accelerated Port Bit Vector Table - APBVT

The L4 destination port number is compared against an internal Accelerated Port Bit Vector Table (GLQF_APBVT registers). The E810 includes a 64K x 1-bit vector for the whole device (all eight LAN ports). An active bit in the table means that the port number is enabled by one (or more) functions. Programming the table is done by the PE interface, as described in Section 11.5.3.18.

For the sake of the APBVT functionality, the destination L4 port number is posted by the extractor logic to word zero in the Field Vector.

Quad Hash Filter - QH

The QH Filter checks for perfect match between selected fields in the packet (called input set) and pre-programmed values in the filter table. The E810 stores some of the QH filters in an internal cache (holding up to 1K filters) while the other filters are kept in host memory (FPM). The maximum number of PE contexts for the device is 256K. These contexts can be allocated in a flexible manner between the functions with a maximum limit per function of 256K minus 1.

Filter Candidate Rules

Packets are forwarded to the PE if they meet the following conditions:

- The receive packet is not rejected for the filter by the exception rules listed in Section 7.10.1.5.
- Receive to a PF — The PE is enabled for the PF by the *PE_ENA* flag in the PFQF_PE_ENA register.
- Receive to a VF — The PE is enabled for the VF and its parent PF by the *PE_ENA* flags in the matched VP/PFQF_PECTL registers.
- Receive to a VSI — The PE is enabled for the VSI by the *PE_ENA* flags in the matched VSIQF_PE_CTL1 register (should be set by add VSI).
- The packet profile is enabled for the VSI group as shown in Table 7-160 (the VSI is associated to a VSI group for the PE by the *VSIG_PE* parameter in the Add VSI admin command). The add VSI admin command PE enable sets *PE_FLTENA* bit in the corresponding VSIQF_PE_CTRL1 CSR.
- The packet's TC is enabled by the PFQF_PE_TC_CTL registers (programmed by the PF driver).
- The packet hits the APBVT table (if required by the *APBVT_ENA* flag in the GLQF_PE_CTL2 register of the matched packet profile). By default, iWARP packets are candidate for the APBVT. If the packet is candidate for the APBVT, hitting the filter is a prerequisite condition for lookup in the QH filter.
- The packet hits the QH filter (if required by the *TO_QH* field in the GLQF_PE_CTL2 register of the matched packet profile). By default setting:
  - iWARP packets must match the QH filter as a criteria for PE hit.
  - Non-unicast RoCEv2 packets are subjected to QH filter check but they are forwarded to the PE regardless of hit or miss.

7.10.9.1 PE Quad Hash Filter (QH filter)

The Quad Hash Filter (QH filter) is aimed to match specific flows with the protocol engine. The E810 maintains some of the Quad Hash filter in internal cache, while the other filters are stored in host memory (FPM). The number of quad hash filters per function is a function of the number of flows supported by that function. The Quad Hash Filter is composed of the following components:

QH Filter Candidate Rules

See Section 7.10.9, "Protocol Engine (PE) Filters".

QH Filter Match Criteria

A packet matches a filter entry if the following packet's parameters match the filter context:

- Input Set fields
- Packet's VSI
- Packet profile

Filter Action

The filter provides a hit/miss indication. In the case of hit, the matched quad hash filter index (QPN) is reported to the PE together with the packet. In case of a miss, the destination queue pair from the packet is reported to the PE with the packet.

Filter Programming

A QH filter is added or removed by the PE (using direct internal interface to the filter logic). As part of the filter parameters, the PE provides the identification fields, the packet's profile, and the QPN.

Note: A filter should be programmed by the PE only if the PF or VF are enabled for the PE and the packet profile is enabled for the VSI group.

Filter Removal

Filters are removed in one of the following options:

- Remove a single filter by the PE using its programming interface.
- PF Reset — Auto-evict all filters of the PF and its VFs following a PF reset. After the reset is completed, or before the PE QH filter is re-enabled, the software driver must clear all the filter entries in the Hash space in the FPM (the Hash space is indicated by PEHSIZE in the Figure 7-33). Only then it can set the *PE_ENA* flag in the PF/VPQF_PE_ENA register.
- VF Reset initiated by the software driver — Auto-evict all filters of the VFs following a VF reset. After the reset is completed, or before the PE QH filter is re-enabled, the parent PF software driver must clear all the filter entries in the Hash space in the FPM (the Hash space is indicated by PEHSIZE in Figure 7-33). Only then it can set the *PE_ENA* flag in the VPQF_PE_ENA register.
- Explicit request by the PE to remove all filters of the function (PF or VF) from the cache. The PE driver initiates a request to clear all filters of a function through the CSR fabric.

QH Table Size

The E810 supports up to 256K Quad Hash contexts for the whole device. These filters are fetched to internal cache that can hold up to 1K filters. The number of supported QH filters is limited by the *PEDSIZE* parameters in the PFQF_PE_CTL register for the PF and VPQF_PE_CTL registers for the VFs. Programming request for a QH filter above these limits is rejected.

QH Filter Context

Each filter entry is consisted of 64 bytes.

Storage in Host Memory

The filters are backed up in the function's private memory (FPM) in buckets as described in Figure 7-33. Filters are stored in the "hash space" at address defined by the hash value that is calculated on the input set (the hash function is Toeplitz using a global key defined by the GLQF_HKEY registers). If this location is occupied (hash collision), the filter is stored in a free location in the "collision space".

The filter in the "hash space" points to all the other filters in the "collision space" that share the same hash. It is organized in a linked list structure. The number of supported buckets (hash space) and the number of supported filters (collision space) are defined per function by the *PEHSIZE* and *PEDSIZE* parameters in the PFQF_PE_CTL and VPQF_PE_CTL registers. Software should clear the hash space in the FPM before filters are programmed. The QH filters of the PF and its VFs are stored in the FPM of the PF.

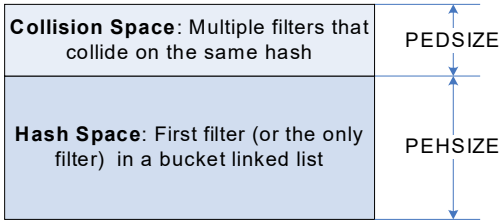


Figure 7-33. Buckets and Collided Filter Entries Space

The actual number of Hash Table entries for the PF is programmed using the PFQF_PE_CTL1 and PFQF_PE_CTL2. Per VF it is programmed through the VPQF_PE_CTL1 and VPQF_PE_CTL2 registers.

7.10.9.1.1 Performance

7.10.9.1.1.1 Programming and Removal Performance Targets

Expected filter programming and removal rate equals to 0.5M filters per second.

7.10.9.1.1.2 Buckets Length Impact on Filter Search Performance

The E810 is able to sustain its maximum packet rate when the matched QH filters are stored in the cache. In the case of miss, the E810 fetches the filters from the FPM (host memory). Host memory latency for fetching the filtering depends on the bucket length. The E810 reports the number of buckets for the function in the PF/VPQF_PECNT_0.*BUCKETCNT*. It also reports the number of programmed filters by the function in the PF/VPQF_PECNT_1 registers. The ratio between these two numbers provides an indication for the QH table distribution and hence for possible long buckets.

On top of it the PE QH table counts each packet positioning in its bucket at reception time. Received packets at a position shorter or equal than GLQF_PE_CTL.*PELONG* are counted by PFQF_PE_CLSN0.*HITSBCNT*, while packets at position longer than *PELONG* are counted by the PFQF_PE_CLSN1.*HITLBCNT*. When GLQF_PE_CTL.*PELONG* equals 0, the functionality is disabled and all packets are counted by PFQF_PE_CLSN1. The counting is qualified by PFQF_PE_ST_CTL CSR.

The *PELONG* threshold applies to all PE packets, including iWARP and RoCEv2. However, for RoCEv2 packet, where there is no QH lookup, the "position in bucket" is always considered as shorter than or equal to GLQF_PE_CTL.*PELONG*.

Received packets that match PE filters at position longer than GLQF_PE_CTL.*PELONG* are reported by the *PELONGB* flag in the receive descriptor. Thus, software can use these parameters as a criteria for the QH table performance. It then could decide to remove some filters that possibly hit performance or flush the whole table.

7.10.10 Hash Functions

The E810 supports several hash functions to be used by the various filters:

- Bucket number of the FD filter and the PE quad hash filter use the Toeplitz scheme with a shared global key defined by the GLQF_HKEY registers.
- The hash filter uses one of the following schemes selected per VSI by the *Hash Scheme* field in the Add VSI command.

— **Toeplitz Hash** — The hash key for the VSI is defined by the VSIQF_HKEY registers (detailed in Section 7.10.10.1).

— **Symmetric Toeplitz Hash** — The hash key is defined as above by the VSIQF_HKEY registers. Symmetric hash is calculated by XOR function between "source" and "destination" tuples providing the same hash signature for Tx and Rx packets of the same "flows" (detailed in Section 7.10.10.2).

— **XOR Hash** — Simple XOR hash function between the DWords of the input set. The input set is padded with zeros if not whole number of DWords. If the hash function is used as an address to a lookup table smaller or equal then 64K entries, a 16-bit hash is used by XOR function of the 16 MS bits and 16 LS bits of the 32-bit hash. If the hash function is used as an address to a lookup table smaller or equal than 256 entries, an 8-bit hash is used by a XOR function of the four bytes of the 32-bit hash.

— **Jenkin's Hash** — This options is provided only for small input sets up to four bytes long (detailed in Section 7.10.11).

The following notations are used to describe the hash functions:

- Ordering is big endian in both bytes and bits. For example, the IP Address 161.142.100.80 translates into 0xA18E6450.
- A "**^**" denotes bit-wise XOR operation of same-width vectors.
- **@x-y** denotes bytes x through y (inclusive) of the incoming packet, where Byte 0 is the first byte of the IP header. In other words, all byte-offsets are considered 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 IPv4 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.

7.10.10.1 Microsoft Toeplitz-Based Hash

The hash uses a random secret key of 416 bits (52 bytes). The key is defined per VSI by the VSIQF_HKEY registers. 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]

Where:

- 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 input[] of length N bytes (8N bits) and a random secret key K of 416 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; // Cyclic shift while the right-most bit of
                                         // K gets the leftmost bit of K
}
return Result;
```

7.10.10.1.1 Pseudo-Code Examples

The following four pseudo-code examples are intended to help clarify exactly how the hash is performed in four cases: IPv4 with and without ability to parse the L4 header, and IPv6 with and without a L4 header.

Table 7-145. Examples of Input Fields for the Hash Function

Packet Type	Hash Input	Hash Result
IPv4 with TCP or UDP	SourceAddress, DestinationAddress, SourcePort, DestinationPort: Input[12] = @12-15, @16-19, @20-21, @22-23	Result = ComputeHash(Input, 12)
IPv4 without L4	SourceAddress, DestinationAddress: Input[12] = @12-15, @16-19	Result = ComputeHash(Input, 8)
IPv6 with TCP or UDP	SourceAddress, DestinationAddress, SourcePort, DestinationPort: Input[12] = @8-23, @24-39, @40-41, @42-43	Result = ComputeHash(Input, 36)
IPv6 without L4	SourceAddress, DestinationAddress: Input[12] = @8-23, @24-39	Result = ComputeHash(Input, 32)

7.10.10.1.2 RSS Verification Suite

This section provides a verification suite used to validate that the hash function is computed according to MSFT nomenclature.

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,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00
```

Table 7-146. 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

The IPv6 Address tuples are only for verification purposes, and might not make sense as a tuple.

Table 7-147. IPv6

Destination Address/Port	Source Address/Port	IPv4 Only	IPv4 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

7.10.10.2 Symmetric Hash

A symmetric hash provides the same value if its respective source and destination fields are swapped. For example, suppose that the hash is done on the frame source and destination IP Addresses. A symmetric hash guarantees that: **hash(src IP, dst IP)** = **hash(dst IP, src IP)**.

The motivation behind symmetric hash is to route frames of a specific "flow" between two nodes to the same receive queue independent of the transmission direction (Tx or Rx).

The symmetric hash is useful for matched pairs of tuples like:

- IP Addresses
- L4 port numbers
- MAC Addresses

Symmetric hash is defined per packet profile by its GLQF_HSYMM registers. For each word with active *SYMM* flag in the GLQF_HSYMM registers, the input set is defined by a XOR function of the words defined by the matched GLQF_HSYMM and GLQF_HINSET registers.

7.10.10.2.1 Symmetric Hash Example for IPv4 with TCP

Non Symmetric input vector is:

```
Input[12] = @12-15, @16-19, @20-21, @22-23
```

Symmetric input vector is:

```
Input[12] = @12-15 ^ @16-19, @12-15 ^ @16-19, @20-21 ^ @22-23, @20-21 ^ @22-23.
```

7.10.11 Receive Filters Admin Commands

7.10.11.1 Set RSS Key (0x0B02)

This command is used to set the RSS Key of a VSI. It can be used by a PF or a VF for VSIs assigned to it. A PF can set the RSS key on behalf of its VFs.

Table 7-148. Set RSS Key Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0B02	Command opcode.
Datalen	4-5	0x34	Length of buffer. Should be equal to 0x34 (hash key size).
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
VSI#	16-17	0x0	**Bit 15:** Valid VSI# Must always be set. Ignored by firmware. **Bits 14:10:** Reserved **Bits 9:0:** VSI Number
Reserved	18-23	0x0	Reserved.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

The command buffer of this command contains the RSS key to set:

Offset	Content
0x0 - 0x27	Standard RSS Key
0x28 - 0x33	Extended hash key. Must be zero for regular RSS.

Table 7-149. Set RSS Key Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0B02	Command opcode.
Datalen	4-5	0x0	No return buffer.
Return Value/VFID	6-7		Return value. The following error values can be returned: ENOENT = If not a valid VSI. EACCES = If the VSI is not owned by this PF/VF.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-23	0x0	Reserved.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

7.10.11.2 Set RSS LUT (0x0B03)

This command is used to set the RSS LUT of a VSI or RSS LUT of the PF or one of the global RSS LUT. This command can be initiated only by the PF. A VF can request its parent PF to configure an RSS LUT for its use via software interface or by the Mailbox AQ. It can be used by a PF or a VF for VSIs assigned to it. A PF can set the RSS LUT on behalf of its VFs.

Table 7-150. Set RSS LUT Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0B03	Command opcode.
Datalen	4-5	64/128/512	Length of buffer. Should be equal to 0x
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
VSI#	16-17		**Bit 15:** Valid VSI# Is set if *Table Type* = VSI. Otherwise, ignored by firmware. **Bits 14:10:** Reserved **Bits 9:0:** VSI Number
Command Flags	18-19		**Bits 0:1:** Table Type 00b = VSI 01b = PF 10b = Global LUT 11b = Reserved **Bits 2:3**: LUT Size For VSI LUT this field is reserved and must be set to 00b. For PF LUT: 00b = 128 01b = 512 10b = 2K 11b = Reserved For Global LUT: 00b = 128 01b = 512 All other values are reserved. **Bits 4:7:** Global LUT Index Relevant only for *Table Type* equals to "Global LUT". Otherwise it must be 0. **Bits 15:8:** Reserved
Reserved	20-23	0x0	Reserved.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

The command buffer of this command contains the LUT to set:

Offset	Content
0x0 - LUT Size	LUT content. Must match the format of the VSIQF_HLUT or PFQF_HLUT registers or GLQF_HLUT registers. For VSI LUT - 64 bytes For PF LUT or Global LUT it is according to the LUT size.

Upon reception of this command, the firmware:

1. Checks if the VSI or the Global LUT belongs to the function that sent the command.
2. Writes the VSIQF_HLUT or PFQF_HLUT registers or GLQF_HLUT registers from the command buffer according to the *Table Type* flag. When a PF table is written, the software might choose to write only the first 128 entries.

Table 7-151. Set RSS LUT Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0B03	Command opcode.
Datalen	4-5		No return buffer.
Return Value/VFID	6-7		Return value. The following error values can be returned: ENOENT = If not a valid VSI. EACCES = If the VSI or the global LUT is not owned by this PF.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-23		Reserved.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

7.10.11.3 Get RSS Key (0x0B04)

This command is used to get the RSS Key of a VSI. It can be used by a PF or a VF for VSIs assigned to it. A PF can query the RSS key of its VFs.

Table 7-152. Get RSS Key Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0B04	Command opcode.
Datalen	4-5	0x34	Length of buffer. Should be equal to 0x34 (has key size) or more.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
VSI#	16-17		**Bit 15:** Valid VSI# Must always be set. Ignored by firmware. **Bits 14:10:** Reserved **Bits 9:0:** VSI Number
Reserved	18-23	0x0	Reserved.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Upon reception of this command, the firmware:

1. Checks if the VSI belongs to the function that sent the command using the VSI_VSI2F[VSI] register.
2. Reads the VSIQF_HKEY[VSI] registers and fill the response buffer.

Table 7-153. Get RSS Key Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0B04	Command opcode.
Datalen	4-5	0x0	0x34
Return Value/VFID	6-7		Return value. The following error values can be returned: ENOENT = If not a valid VSI. EACCES = If the VSI is not owned by this PF/VF.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-23	0x0	Reserved.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

The response buffer of this command contains the RSS key currently set:

Offset	Content
0x0 - 0x27	Standard RSS Key
0x28 - 0x33	Extended hash key. Must be zero for regular RSS.

7.10.11.4 Get RSS LUT (0x0B05)

This command is used to get the RSS LUT of a VSI or RSS LUT of the PF or one of the global RSS LUT. This command can be initiated only by the PF. A VF can request its parent PF to configure an RSS LUT for its use via software interface or by the Mailbox AQ. It can be used by a PF or a VF for VSIs assigned to it. A PF can query the RSS LUT of its VFs.

Table 7-154. Get RSS LUT Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0B05	Command opcode.
Datalen	4-5	64/512	Length of buffer. Should be equal to 512 or more for a PF table, or 64 or more for a VSI table.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.

Table 7-154. Get RSS LUT Command [continued]

Name	Byte.Bit	Value	Remarks
VSI#	16-17		**Bit 15:** Valid VSI# Is set if *Table Type* = VSI. Otherwise, ignored by firmware. **Bits 14:10:** Reserved **Bits 9:0:** VSI Number
Command Flags	18-19		**Bits 0:1:** Table Type 00b = VSI 01b = PF 10b = Global LUT 11b = Reserved **Bits 2:3**: Reserved Must be set to zero. **Bits 4:7:** Global LUT Index Relevant only for *Table Type* equals to "Global LUT". Otherwise it must be 0. **Bits 15:8:** Reserved
Reserved	20-23	0x0	Reserved.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Upon reception of this command, the firmware:

1. Checks if the VSI or the Global LUT belongs to the function that sent the command.
2. Reads the VSIQF_HLUT or PFQF_HLUT registers and fill the response buffer according to the *Table Type* flag. When a PF table is read, the entire 512 entries are read.

Table 7-155. Get RSS LUT Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0B05	Command opcode.
Datalen	4-5	0x0	0x2
Return Value/VFID	6-7		Return value. The following error values can be returned: ENOENT = If not a valid VSI. EACCES = If the VSI is not owned by this PF.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-23	0x0	Reserved.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

The response buffer contains the LUT currently set:

Offset	Content
0x0 - LUT Size	LUT content as read from f the VSIQF_HLUT or PFQF_HLUT registers or GLQF_HLUT registers. For VSI LUT - 64 bytes For PF LUT or Global LUT it is according to the LUT size.

7.10.12 Filter Clearing Commands and Flows

7.10.12.1 Clear FD Table (0x0B06)

This command is used to clear the FD filters associated with a given PF or a specific VSI. Upon reception of this command, the firmware must execute the flow described in Section 7.10.12.1.1.

Note: As part of a PF reset processing, the same flow is executed by the firmware for the PF being reset, without the need for software to send an AQC.

Table 7-156. Clear FD Table Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0B06	Command opcode.
Reserved	4-5		Reserved.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Clear_type	16		Clear Type 0x0 = Reserved 0x1 = VSI 0x2 = PF 0x3-0xFF = Reserved
Reserved	17		Reserved.
VSI_index	18-19		VSI Index This field is not applicable and should be set to 0x0 if function type is PF. ***Note:*** Inapplicable MSBs should always be set to 0x0.
Reserved	20-31	0x0	Reserved.

Table 7-157. Clear FD Table Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0B06	Command opcode.
Reserved	4-5		Reserved.
Return Value/VFID	6-7		Return value.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Clear_type	16		Clear Type 0x0 = Reserved 0x1 = VSI 0x2 = PF 0x3-0xFF = Reserved
Reserved	17		Reserved.

Table 7-157. Clear FD Table Response [continued]

Name	Byte.Bit	Value	Remarks
VSI_index	18-19		VSI Index This field is not applicable and should be set to 0x0 if function type is PF. ***Note:*** Inapplicable MSBs should always be set to 0x0.
Reserved	20-31	0x0	Reserved.

7.10.12.1.1 FD Table Clearing Flow

This section describes the flow, executed by the EMP firmware, for clearing the FD filters associated with a given function. The flow can be either initiated by the EMP firmware, for PF reset events, or by software using the relevant AQC.

Clearing the FD table is composed of the following main steps:

1. Block new inputs (searches and configuration) associated with the entity being cleared.
2. Wait for any ongoing/"in-flight" operations associated with the entity being cleared complete.
3. Clear/remove the table entries associated with the cleared entity.
4. Fix the filter counters so they correspond to the new filter amounts.

7.10.12.1.1.1 Clearing the FD Filter Flow

1. Set the following CSR fields:
   - GLQF_FDPIPE_CLR.*DIS_CFG*
   - GLQF_FDPIPE_CLR.*DIS_FLT*
   - GLQF_FDPIPE_CLR.*FD_BUSY*
   - GLQF_FDPIPE_CLR.*VM_VF_TYPE* to the *Clear_type* operand value.
   - GLQF_FDPIPE_CLR.*PF_NUM* to the PF index.
   - GLQF_FDPIPE_CLR.*VM_VF_NUM* to the *VSI_index* operand value.

   Note: As a response to the CSR write, the hardware blocks new search requests and new configuration requests for the cleared entity (PF or VSI). It does so by clearing the following flags:

   For PF entity, it clears the PFQF_FD_ENA.*FD_ENA* flag.

   For VSI entity, it clears the *FLT_ENA*, *CFG_ENA*, and *EVICT_ENA* flags in the VSIQF_FD_CTL1 register.

   Once there are no pending searches and configuration requests in the pipe, the *FD_BUSY* flag is cleared. This flow is implemented by initiating a "mini marker" in the post (from extractor input till end of FLU). Following a marker done indication, the *FD_BUSY* flag is cleared.
2. The EMP firmware pulls the status of GLQF_FDPIPE_CLR.*FD_BUSY* flag until it is cleared by the hardware.

   Note: At this stage, no additional inputs associated with the cleared entity are allowed to enter the pipe and there are no ongoing operations associated with the cleared entity.

3. The EMP firmware sets the following CSR fields:
   - GLQF_FDTABLE_CLR.*FD_CLEAR*
   - GLQF_FDTABLE_CLR.*FD_BUSY*
   - GLQF_FDTABLE_CLR.*VM_VF_TYPE* to the *Clear_type* operand value.
   - GLQF_FDTABLE_CLR.*VM_VF_NUM* to the *VSI_index* operand value.

   Note: As a response to the CSR write, the hardware invalidates all the filters of the requested entity and clears the GLQF_FDTABLE_CLR.*FD_BUSY* flag.
4. The EMP firmware pulls the status of GLQF_FDTABLE_CLR.*FD_BUSY* flag until it is cleared by the hardware.

   Note: At this stage, the filter entries associated with the cleared entity are cleared/invalidated.
5. In case of a VSI entity clear:
   a. The EMP firmware records the value of the following fields:
      - VSIQF_FD_CNT[VSI_index].*FD_GCNT*
      - VSIQF_FD_CNT[VSI_index].*FD_BCNT*

   b. The EMP firmware clears VSIQF_FD_CNT[VSI_index].*FD_GCNT* and VSIQF_FD_CNT[VSI_index].*FD_BCNT* by writing zero to the relevant fields.

   c. The EMP firmware writes the values, recorded instep a above, to PFQF_FD_SUBTRACT[PF_index].*FD_GCNT* and PFQF_FD_SUBTRACT[PF_index].*FD_BCNT*.
6. This operation subtract the filter count of the removed VSI from the PFQF_FD_CNT and GLQF_FD_CNT registers.

   Note: For PF reset case, all the counters associated with the reseted PF's VSIs are zeroed by software as part of the driver initialization. Therefor they do not need to be addressed as part of the clear flow.

   In case the filters associated with a PF are being cleared not as part of a PF reset event (that is, the clearing is initiated by the software), the software can follow the steps specified above for fixing the counters' values. If software does not decrement the counters, the counters' values will not correspond to the actual amount of filters stored in the FD storage.
7. The EMP firmware clears the GLQF_FDPIPE_CLR.*DIS_CFG* and GLQF_FDPIPE_CLR.*DIS_FLT* bits, and reports the completion of the clear flow to software.

   Note: At this stage, the PF/VSI is still not enabled yet, as hardware cleared *FD_ENA*(PF) or *FLT_ENA*/*CFG_ENA*/*EVICT_ENA* (VSI) as described at the note in Step 1.
8. The software can re-enable the FD for the VSI/PF by setting the applicable flags:
   - For PF entity, it sets the PFQF_FD_ENA.*FD_ENA* flag.
   - For VSI entity, it sets the *FLT_ENA*, *CFG_ENA*, and *EVICT_ENA* flags in the VSIQF_FD_CTL1 register.

7.10.12.2 QH Table Clearing

7.10.12.2.1 QH Table Clearing Flow

This section describes the flow for clearing the PE QH table for a specific PF/VF.

Note: This flow is not expected to be executed by software, and clearing of the QH filters is done solely by the PE firmware in functional environment. The option for software to clear the QH filters, mentioned in this section is therefore for debug purposes only.

Note: Removing the filters from the FPM is the responsibility of software.

The same flow can also be processed by the PE firmware as part of an internal table removal process. In this case, the instance index #0 for CSRs that appear in this flow should be replaced by instance index #1.

1. The software/firmware sets the following CSR fields:
   - GLQF_PEPIPE_CLR[0].*DIS_FLT*
   - GLQF_PEPIPE_CLR[0].*PE_BUSY*
   - GLQF_PEPIPE_CLR[0].*PF_VF_TYPE* to the *Clear_type* operand value.
   - GLQF_PEPIPE_CLR[0].*PF_VF_NUM* to the *VF_index* operand value.

   Note: As a response to the CSR write, the hardware blocks new search requests and new configuration requests for the entity being cleared by clearing the *PE_ENA* flag in the relevant PF/VPQF_PE_ENA registers.

   Once there are no pending searches and pending configuration requests, the hardware clears the GLQF_PEPIPE_CLR[0].*PE_BUSY* flag. This flow is implemented by initiating a "mini marker" in the post (from extractor input till end of FLU). Following a marker done indication, the *PE_BUSY* flag is cleared.
2. The software/firmware polls the status of GLQF_PEPIPE_CLR[0].*PE_BUSY* flag until it is cleared.

   Note: At this stage, no additional inputs associated with the cleared entity are allowed to enter the pipe and there are no ongoing operations associated with the cleared entity.
3. The software/firmware sets the following CSR fields:
   - GLQF_PETABLE_CLR[0].*PE_CLEAR*
   - GLQF_PETABLE_CLR[0].*PE_BUSY*
   - GLQF_PETABLE_CLR[0].*PF_VF_TYPE* to the *Clear_type* operand value.
   - GLQF_PETABLE_CLR[0].*PF_VF_NUM* to the *VF_index* operand value.

   Note: As a response to the CSR write, the hardware clears the table pointers associated with the cleared function and then clear the GLQF_PETABLE_CLR[0].*PE_BUSY* flag.
4. The software/firmware polls the status of GLQF_PETABLE_CLR[0].*PE_BUSY* flag until it is cleared.

   Note: At this stage, the logical tables holding the QH filters for the cleared function are cleared.
5. The software/firmware clears the filter counters associated to the function being cleared:
   a. For a VF Clear, clear following fields:
      - VPQF_PECNT1[VF_index].*FLTCNT*
      - VPQF_PECNT0[VF_index].*BUCKETCNT*

- VPQF_PE_FLHD[VF_index].*FLHD*

Note: Since VF counter values are not accumulated in the associated PF counter, there is no need to touch the PF counter in that case.

b. For a PF Clear, clear the following associated fields:

- PFQF_PECNT1[PF_index].*FLTCNT*
- PFQF_PECNT0[PF_index].*BUCKETCNT*
- PFQF_PE_FLHD[PF_index].*FLHD*

6. The software/firmware clears the GLQF_PEPIPE_CLR[0].*DIS_FLT* bit.
7. Software steps to re-enable the filter:
   a. Clear the filter entries in the Hash space in the FPM (the Hash space is indicated by PEHSIZE in the Figure 7-33).
   b. Set the PF/VPQF_PE_ENA.*PE_ENA* flags that were cleared in the clear process.

   This step is similar to any PF/VF initialization flow.

7.10.13 Default Extractor Configuration

This section describe the default configuration of the packet profile tables as well as the field vector extraction tables in the NVM.

Note: The device is programmed to the NVM default when the driver does not load any package. Therefore, the configuration described in this section is provided as sample configuration, while the driver might load different configuration package during initialization.

7.10.13.1 Profiles, Field Vector, and Input Sets

Table 7-158 through Table 7-160 show the default setting of the field vector and input set for the FD filter and Hash filter selected by control domain zero. And also the default setting of the PE quad hash filter.

Table 7-158. Default Hash filter Field Vector Configuration

Profile ID	Description	Hash Filter Field Vector	Possible Symmetric Fields
Miss the Filter - Combination of packet types and flags that do no match any profile results with a default profile index.			
0	No Match	None	None
1	Reserved for Bypass the filter	None	None
IPv4 Packets			
2	NonF IPv4, TCP **and** VSIG[2]=0	IP-S, IP-D, TCP-S, TCP-D	IP-S, IP-D, TCP-S, TCP-D
3	NonF IPv4, UDP-(incl. RoCEv2) **and** VSIG[2]=0	IP-S, IP-D, UDP-S, UDP-D	IP-S, IP-D, UDP-S, UDP-D
4	NonF IPv4, Payload/Other **or** Fragmented IPv4 **or** VSIG[2]=1	IP-S, IP-D	IP-S, IP-D
IPv4 Tunneled Packets (fields are inner header fields)			
5	NonF IPv4, TCP **and** VSIG[2]=0	IP-S, IP-D, TCP-S, TCP-D	IP-S, IP-D, TCP-S, TCP-D
6	NonF IPv4, UDP-(incl. RoCEv2) **and** VSIG[2]=0	IP-S, IP-D, UDP-S, UDP-D	IP-S, IP-D, UDP-S, UDP-D

Table 7-158. Default Hash filter Field Vector Configuration [continued]

Profile ID	Description	Hash Filter Field Vector	Possible Symmetric Fields
7	NonF IPv4, Payload/Other **or** Fragmented IPv4 **or** VSIG[2]=1	IP-S, IP-D	IP-S, IP-D
IPv6 Packets			
8	NonF IPv6, TCP **and** VSIG[1]=0	IP-S, IP-D, TCP-S, TCP-D	IP-S, IP-D, TCP-S, TCP-D
9	NonF IPv6, UDP-(incl. RoCEv2) **and** VSIG[1]=0	IP-S, IP-D, UDP-S, UDP-D	IP-S, IP-D, UDP-S, UDP-D
10	NonF IPv6, Payload/Other **or** Fragmented IPv6 **or** VSIG[1]=1	IP-S, IP-D	IP-S, IP-D
IPv6 Tunneled Packets (fields are inner header fields)			
11	NonF IPv6, TCP and VSIG[1]=0	IP-S, IP-D, TCP-S, TCP-D	IP-S, IP-D, TCP-S, TCP-D
12	NonF IPv6, UDP-(incl. RoCEv2) **and** VSIG[1]=0	IP-S, IP-D, UDP-S, UDP-D	IP-S, IP-D, UDP-S, UDP-D
13	NonF IPv6, Payload/Other **or** Fragmented IPv6 **or** VSIG[1]=1	IP-S, IP-D	IP-S, IP-D

Notes:

- All PTGs above are derivatives of tunneled and non-tunneled packet types. For tunneled packets, it is expected that the OAM flag is inactive. Active OAM will yield "no Match" PTG.
- By default, settings the profiles for the packet types depends on the VSIG(XLT2), as follows:
  - VSIG 0x00 — The filter is not enabled (all packet types Miss).
  - VSIG bit 1 = 1 — IPv6 packets use IP tuples only for hash.
  - VSIG bit 2 = 1 — IPv4 packets use IP tuples only for hash.

  Although the different VSIGs are supported in the NVM, the NVM generate only the VSIG=0x80 case.
- In the hash filters, all fields must be "left shift" in the Input set to comply with Toeplitz RSS hash.

Table 7-159. Default FD Filters Profiles and Key Structure (Loaded from NVM)

Profile ID	PTG(XLT1) and PFLAG_G	FD Filter Key	Possible Swapped Field (FD Filter)
Miss the Filter - Combination of packet types and flags that do no match any profile results with a default profile index.			
0	No Match	None	None
1	Reserved for Bypass the filter	None	None
IPv4 Packets			
2	NonF IPv4, TCP, SYN No ACK **and** VSIG[2]=0 **and** VSIG[3]=0	IP-D, TCP-D	IP-S, IP-D, TCP-S, TCP-D
3	NonF IPv4, TCP **and** VSIG[2] = 0 **or** (NonF IPv4, TCP, SYN **and** VSIG[3]=1)	IP-S, IP-D, TCP-S, TCP-D	IP-S, IP-D, TCP-S, TCP-D
4	NonF IPv4, UDP-(incl. RoCEv2) **and** VSIG[2]=0	IP-S, IP-D, UDP-S, UDP-D	IP-S, IP-D, UDP-S, UDP-D
5	NonF IPv4, ESP **and** VSIG[2]=0 **and** VSIG[0]=0	IP-S, IP-D, SPI	IP-S, IP-D
6	NonF IPv4, SCTP **and** VSIG[2]=0	IP-S, IP-D, SCTP-S, SCTP-D, SCTP-VTag	IP-S, IP-D, SCTP-S, SCTP-D
7	NonF IPv4, Payload/Other **or** VSIG[2]=1 **or** (NonF IPv4, ESP **and** VSIG[0]=1)	IP-S, IP-D	IP-S, IP-D
8	Fragmented IPv4	IP-S, IP-D	IP-S, IP-D
IPv4 Tunneled Packets (fields are inner fields)			
9	NonF IPv4, TCP, SYN No ACK **and** VSIG[2]=0 **and** VSIG[3]=0	IP-D, TCP-D	IP-S, IP-D, TCP-S, TCP-D
10	NonF IPv4, TCP **and** VSIG[2]=0 **or** (NonF IPv4, TCP, SYN **and** VSIG[3]=1)	IP-S, IP-D, TCP-S, TCP-D	IP-S, IP-D, TCP-S, TCP-D

Table 7-159. Default FD Filters Profiles and Key Structure (Loaded from NVM) [continued]

Profile ID	PTG(XLT1) and PFLAG_G	FD Filter Key	Possible Swapped Field (FD Filter)
11	NonF IPv4, UDP-(incl. RoCEv2) **and** VSIG[2]=0	IP-S, IP-D, UDP-S, UDP-D	IP-S, IP-D, UDP-S, UDP-D
12	NonF IPv4, ESP **and** VSIG[2]=0 **and** VSIG[0]=0	IP-S, IP-D, SPI	IP-S, IP-D
13	NonF IPv4, SCTP **and** VSIG[2]=0	IP-S, IP-D, SCTP-S, SCTP-D, SCTP-VTag	IP-S, IP-D, SCTP-S, SCTP-D
14	NonF IPv4, Payload/Other **or** VSIG[2]=1 **or** (NonF IPv4, ESP **and** VSIG[0]=1)	IP-S, IP-D	IP-S, IP-D
15	Fragmented IPv4	IP-S, IP-D	IP-S, IP-D
IPv6 Packets			
16	NonF IPv6, TCP, SYN No ACK **and** VSIG[1] 0 **and** VSIG[3]=0	IP-D, TCP-D	IP-S, IP-D, TCP-S, TCP-D
17	NonF IPv6, TCP **and** VSIG[1]=0 **or** (NonF IPv6, TCP, SYN **and** VSIG[3]=1)	IP-S, IP-D, TCP-S, TCP-D	IP-S, IP-D, TCP-S, TCP-D
18	NonF IPv6, UDP-(incl. RoCEv2) **and** VSIG[1]=0	IP-S, IP-D, UDP-S, UDP-D	IP-S, IP-D, UDP-S, UDP-D
19	NonF IPv6, ESP **and** VSIG[1]=0 **and** VSIG[0]=0	IP-S, IP-D, SPI	IP-S, IP-D
20	NonF IPv6, SCTP **and** VSIG[1]=0	IP-S, IP-D, SCTP-S, SCTP-D, SCTP-VTag	IP-S, IP-D, SCTP-S, SCTP-D
21	NonF IPv6, Payload/Others **or** VSIG[1]=1 **or** (NonF IPv6, ESP **and** VSIG[0]=1)	IP-S, IP-D	IP-S, IP-D
22	Fragmented IPv6	IP-S, IP-D	IP-S, IP-D
IPv6 Tunneled Packets (fields are inner fields)			
23	NonF IPv6, TCP, SYN No ACK **and** VSIG[1]=0 **and** VSIG[3]=0	IP-D, TCP-D	IP-S, IP-D, TCP-S, TCP-D
24	NonF IPv6, TCP **and** VSIG[1]=0 **or** (NonF IPv6, TCP, SYN **and** VSIG[3]=1)	IP-S, IP-D, TCP-S, TCP-D	IP-S, IP-D, TCP-S, TCP-D
25	NonF IPv6, UDP-(incl. RoCEv2) **and** VSIG[1]=0	IP-S, IP-D, UDP-S, UDP-D	IP-S, IP-D, UDP-S, UDP-D
26	NonF IPv6, ESP **and** VSIG[1]=0 **and** VSIG[0]=0	IP-S, IP-D, SPI	IP-S, IP-D
27	NonF IPv6, SCTP **and** VSIG[1]=0	IP-S, IP-D, SCTP-S, SCTP-D, SCTP-VTag	IP-S, IP-D, SCTP-S, SCTP-D
28	NonF IPv6, Payload/Others **or** VSIG[1]=1 **or** (NonF IPv6, ESP **and** VSIG[0]=1)	IP-S, IP-D	IP-S, IP-D
29	Fragmented IPv6	IP-S, IP-D	IP-S, IP-D
GTP Tunneling			
30	IPv4, GTP-C, TEID	IP-S, IP-D, UDP-S,UDP-D, TEID	IP-S, IP-D, UDP-S, UDP-D
31	IPv4, GTP-C, No TEID	IP-S, IP-D, UDP-S,UDP-D	IP-S, IP-D, UDP-S, UDP-D
32	IPv4, GTP-U, service = 0xFF	IP-S, IP-D, UDP-S,UDP-D, TEID	IP-S, IP-D, UDP-S, UDP-D
33	IPv4, GTP-U, service != 0xFF	IP-S, IP-D, UDP-S,UDP-D, TEID	IP-S, IP-D, UDP-S, UDP-D
34	IPv6, GTP-C, TEID	IP-S, IP-D, UDP-S,UDP-D, TEID	IP-S, IP-D, UDP-S, UDP-D
35	IPv6, GTP-C, No TEID	IP-S, IP-D, UDP-S,UDP-D	IP-S, IP-D, UDP-S, UDP-D
36	IPv6, GTP-U, service = 0xFF	IP-S, IP-D, UDP-S,UDP-D, TEID	IP-S, IP-D, UDP-S, UDP-D
37	IPv6, GTP-U, service != 0xFF	IP-S, IP-D, UDP-S,UDP-D, TEID	IP-S, IP-D, UDP-S, UDP-D

Table 7-159. Default FD Filters Profiles and Key Structure (Loaded from NVM) [continued]

Profile ID	PTG(XLT1) and PFLAG_G	FD Filter Key	Possible Swapped Field (FD Filter)
L2 Packet Types			
38	NSH	Service path header and Index	
39	ARP	TPA	SPA, TPA
40	Other L2 packets	L2 EtherType	

Notes:

- All PTGs above are derivatives of tunneled and non-tunneled packet types. For tunneled packets, it is expected that the OAM flag is inactive. Active OAM will yield "no Match" PTG.
- By default, settings the profiles for the packet types depends on the VSIG(XLT2), as follows:
  - VSIG 0x00 — The filter is not enabled (all packet types Miss).
  - VSIG bit 0 = 1 — Mapping of the packet types other than ESP is according to this table. ESP packets are mapped to IP, Other.
  - VSIG bit 1 = 1 — Use only IP tuples for all IPv6 packets.
  - VSIG bit 2 = 1 — Use only IP tuples for all IPv4 packets.
  - VSIG bit 3 = 1 — Treat TCP/SYN (with/without ACK) packets as TCP packets.

  Although the different VSIGs are supported in the NVM. The NVM generate only the VSIG=0x80 case.
- In the FD filters this is a possible setting but not mandatory for all fields to be "left shifted" in the Input set. In those fields that the mask option is used, it is recommended to locate them in the same location in the Input set for all packet profiles. This way, the mask registers utilization is optimized.

Table 7-160. Default PE Filter Profiles and Input Sets (Loaded from NVM)

Profile ID	PTG and PFLAG_G	PE Filter FV and Input Set	Additional Match Criteria
Miss the Filter - Combination of packet types and flags that do no match any profile results with a default profile index.			
0	No Match	None	None
1	Reserved for Bypass the filter	None	None
TCP iWARP Packets			
2	NonF IPv4, TCP, SYN No ACK (Unicast)	D-MAC+VLAN, IP-D, TCP-D	Packet must match the APBVT to be a candidate for the QH filter. Then, it should match the QH filter to be a candidate for the PE.
3	NonF IPv4, TCP (Unicast)	D-MAC+VLAN, IP-S, IP-D, TCP-S, TCP-D	
4	NonF IPv6, TCP, SYN No ACK (Unicast)	D-MAC+VLAN, IP-D, TCP-D	
5	NonF IPv6, TCP (Unicast)	D-MAC+VLAN, IP-S, IP-D, TCP-S, TCP-D	
UDP Packets			
6	NonF IPv4, UDP, Unicast	D-MAC+VLAN, Dest IP, Dest UDP port	Packet must match the APBVT to be a candidate for the QH filter. Then, it should match the QH filter to be a candidate for the PE.
7	NonF IPv6, UDP, Unicast	D-MAC+VLAN, Dest IP, Dest UDP port	
8	NonF IPv4, UDP, Multicast	D-MAC+VLAN, Dest IP, Dest UDP port	
9	NonF IPv6, UDP, Multicast	D-MAC+VLAN, Dest IP, Dest UDP port	
RoCEv2 Packets			
10	NonF IPv4, UDP-RoCEv2, Unicast	Dest Queue Pair(5)	Packet bypass both APBVT and QH filter.
11	NonF IPv6, UDP-RoCEv2, Unicast	Dest Queue Pair(5)	

Table 7-160. Default PE Filter Profiles and Input Sets (Loaded from NVM) [continued]

Profile ID	PTG and PFLAG_G	PE Filter FV and Input Set	Additional Match Criteria
12	NonF IPv4, UDP-RoCEv2, non-Unicast	D-MAC+VLAN, Dest IP, Dest Queue Pair(5)	Packet bypass the APBVT. It is forwarded to the PE regardless if it matches the QH filter or not.
13	NonF IPv6, UDP-RoCEv2, non-Unicast	D-MAC+VLAN, Dest IP, Dest Queue Pair(5)	

Notes:
- All PTGs above are derivatives of non-tunneled packet types. Tunneled packets would miss all packet profiles.
- By default, settings the profiles for the packet types depends on the VSIG, as follows:
  - VSIG 0x00 — PE is not enabled (all packet types Miss).
  - VSIG bit 0 = 1— TCP packets are enabled.
  - VSIG bit 1 = 1 — UDP (unicast and multicast) are enabled.
  - VSIG bit 2 = 1— Unicast RoCEv2 are enabled.
  - VSIG bit 3 = 1 — Multicast RoCEv2 are enabled.
- The match criteria in this column are set by the GLQF_PE_CTL2 registers per packet profile.
- In case of RoCE the *Queue Pair* field width is 24 bits.
- All "Unicast", "non-Unicast", and "Multicast" notations used in this table relate to the MAC DA.

Note: Whenever a partial field is needed (sub-word), a mask should be applied to mask out the non-relevant bits of the word. There are 16 GLQF_PEMASK registers. Each register contains a mask and the word it can be applied to. There is a GLQF_PEMASK_SEL register per profile. GLQF_PEMASK_SEL selects the set of GLQF_PEMASK registers that are be applied when the associated profile is selected.

7.11 Packages and Configuration

7.11.1 Introduction

A host system initializes many programmable aspects of the E810 packet processing pipeline through the use of AQ commands and **Packages**. A package contains low-level configuration data representing a broad category packet handling. For example, a package can be tailored to match the networking capabilities of a specific operating system.

Packages primarily contain static configuration information applied during initialization, and only one package can be active at a time. Firmware loads an initial package from NVM, which the host can override with a Download Package command. The E810 supports limited runtime alteration of the current package using the Update Package command. For example, a PF driver might use Update Package to add or remove Packet Profiles dynamically.

Packages have names and version numbers according to the Version Numbers convention in Section 7.11.4. The E810 devices are factory-configured with a default package named DEFAULT (case sensitive). By using the version number, the host can check for driver vs. package compatibility.

A package is a composite of smaller, often interdependent data structures, each mapping to a specific hardware block. A package provides the means to keep these interdependent structures and associated metadata together as an intact unit. For example, a package might contain configuration data for both the Parse Graph CAM and Parser Instruction Memory.

Off-line compiler tools typically produce packages. At runtime, the host can dynamically create package updates to alter the configuration of the current package. In either case, the data structures follow the format specified in this section.

7.11.2 Overriding the Default Package

Using the Download Package command (Section 7.11.9.1), the host system can override the default package loaded from NVM by firmware. The Download Package command changes the configuration of the packet processing pipeline, but does not modify the NVM. A subsequent Core Reset, Global Reset, or power cycle causes firmware to reload the default package and therefore discard the host's custom package. In this case, the host must re-issue another Download Package command.

In the pre-boot environment, the host operating system has not yet loaded and has no opportunity to override the default package. Deployment of a custom package in the pre-boot environment requires either a custom NVM image or host BIOS to issue the Download Package command.

7.11.3 Endianness

Unless otherwise specified, all values in a package use little-endian memory layout. When serialized to a storage device, bytes are placed starting from lowest address to highest address. See Table 7-161.

Table 7-161. Little-Endian Memory Layout for All Value Fields

31 24	23 16	15 8	7 0	Byte Offset
char 15 (Highest Address)	char 14	char 13	char 12	12
int 0				8
short 1		short 0		4
char 3	char 2	char 1	char 0 (Lowest Address)	0

7.11.4 Version Numbers

To convey version and compatibility information, packages use the Major.Minor.Update.Draft (M.m.u.d) convention described in this section.

Consider a hypothetical version 3.0.1.4

Table 7-162. Version Number Fields for Example Version 3.0.1.4

Digit	Name	Description
3	Major	The most significant version number. Configurations with differing Major version numbers might not be compatible with each other.
0	Minor	Within a Major version, the Minor version number indicates compatibility with less-then-or-equal Minor version numbers. A new Minor version number often indicates the addition of new features.
1	Update	A bug fix or refactoring for an existing Major.Minor version.
4	Draft	For internal use only and used to distinguish developmental versions. Shipping configurations should always have a 0 Draft digit.

Each part of the version number is stored in a byte, for a total of four bytes. When stored in memory, the Major number resides in the lowest address byte up to Draft number in the highest address byte. Version numbers are stored as unsigned integer bytes, not as UTF-8 characters. See Table 7-163 for an example.

Table 7-163. Version Number Encoding for Example Version 3.0.1.4

Address	Value	Description
0x1000	0x03	Major
0x1001	0x00	Minor
0x1002	0x01	Update
0x1003	0x04	Draft

7.11.5 Package Format

A package contains a nested hierarchy of structures. In the outermost layer, the package contains an 8-byte fixed-length Package Header. The Package Header contains a version number for the format of the package itself, and a field specifying the number of segments contained in the package. Following those fields, a Segment Table provides the byte offset from the start of package file to each segment in the file. See Figure 7-34.

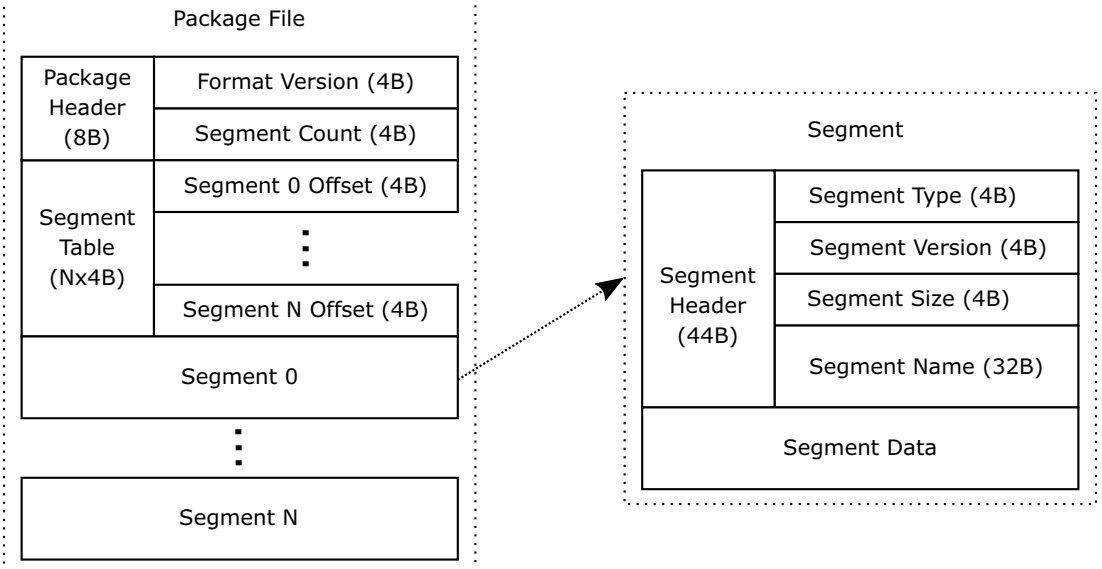


Figure 7-34. Basic Structure of a Package

Each Segment begins with a Segment Header, followed by segment specific data. Segments can contain any amount of data as appropriate for the specified Segment Type.

Table 7-164 describes each field of the Package Header in detail. The Segment Table is sorted from smallest to largest Segment Offset value. All segments start with the Segment Header as shown in Figure 7-34.

Table 7-164. Package Header

Field		Size (Bytes)	Offset	Description
Package Format Version		4	0	M.m.u.d style format for the package itself. The version number follows the convention described in Section 7.11.4. This document describes package format 1.0.1.0.
Segment Count		4	4	Number of segments in the segment table. A value of 0 is reserved.
Segment Table	Segment 0 Offset	4	8	Byte offset from the start of the package file to the first byte of the first segment. Values less than 12 are reserved.
	...			Offsets of all subsequent segments. The Segment Table is sorted from smallest to largest offset, so each Segment Offset value must be at least 44 bytes (Segment Header size) greater than the previous offset.

Table 7-165 describes the fields of the Segment Header in detail.

Table 7-165. Segment Header

Field	Size (Bytes)	Offset	Description
Segment Type	4	0	Type value identifying this segment.
Segment Format Version	4	4	M.m.u.d style format for the package itself. The version number follows the convention described in Section 7.11.4.
Segment Size	4	8	Size in bytes of this segment, including the Segment Header. Values less than 44 bytes are reserved.
Segment Identifier	32	12	Free form (UTF-8 by convention) null-terminated 31-byte name of the segment. The 32nd byte of the name must be a binary 0.

Table 7-166 defines Segment Type numbers relevant to the E810.

Table 7-166. Segment Type Numbers

Segment Type Number	Description	Section Reference
0x0	Reserved	
0x1	Global Metadata Segment	7.11.5.1
0x2	Package Notes Segment	7.11.5.2
0x3-0xF	Reserved	
0x10	E810 Configuration Segment	7.11.5.3

7.11.5.1 Global Metadata Segment

The Global Metadata Segment contains information about the entire package.

Table 7-167 describes the fields of the Global Metadata Segment in detail.

Table 7-167. Global Metadata Segment

Field		Size (Bytes)	Offset	Description
Segment Header	0x00000001	4	0	This segment has a Type value of 1.
	0x00000001	4	4	This document describes Version 1.0.0.0 of this segment.
	0x00000054	4	8	This segment has a length of 84 bytes.
	"Global Metadata"	32	12	This segment has case sensitive name "Global Metadata" without the quote characters. Bytes after the string are padded with binary 0 to full 32-byte length.
Segment Data	Package Version	4	44	Version of the entire Package file as specified by a *version* statement at global scope in an assembly source file. Software tools can use 255.255.255.255 as a reserved value meaning the version is unknown.
	Reserved	4	48	This value is reserved for the E810 and newer devices.
	Package Name	32	52	Free form (UTF-8 by convention) null-terminated 31-byte name of the entire package as specified by a *name* statement at global scope in an assembly source file. Software tools reserve a zero length string (first byte is binary 0) to mean the name is unknown. Software tools always set the 32nd byte of the name to a binary 0.

7.11.5.2 Package Notes Segment

The Package Notes Segment contains free form UTF-8 text about this package. The content of the notes segment is informational only. The notes segment does not affect the configuration of any hardware device.

Table 7-168 shows the content of the Package Notes Segment.

Table 7-168. Package Notes Segment

Field		Size (Bytes)	Offset	Description
Segment Header	0x00000002	4	0	This segment has a Type value of 2.
	0x00000001	4	4	This document describes Version 1.0.0.0 of this segment.
	<variable>	4	8	This segment has variable length.
	"Notes"	32	12	This segment has case sensitive name "Notes" without the quote characters. Bytes after the string are padded with binary 0 to full 32-byte length.
Segment Data	UTF-8 content	<variable>	44	UTF-8 text for informational purposes only. Line feed style is '\n'. The final note in the notes segment always ends with '\0'.

7.11.5.3 E810 Configuration Segment

Configuration data for the E810 resides in the E810 Configuration Segment with Segment Type value 0x10. Figure 7-35 shows an overview of this segment.

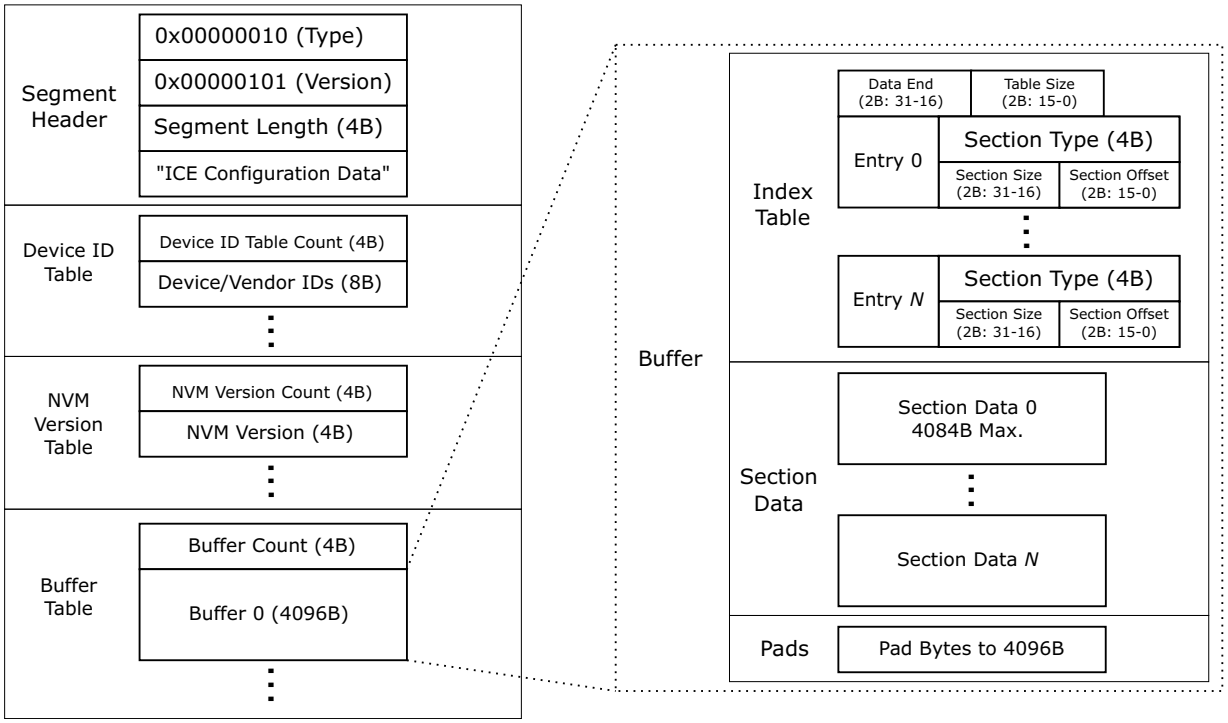


Figure 7-35. Overview of the E810 Configuration Segment

Table 7-169 shows the layout of this segment.

Table 7-169. E810 Configuration Segment

Field		Size (Bytes)	Description
Segment Header	0x00000010	4	This segment has a Type value of 0x10.
	0x00000001	4	This document describes Version 1.0.1.0 of this segment.
	<variable>	4	This segment has variable length.
	"ICE Configuration Data"	32	This segment has case sensitive name "ICE Configuration Data" without the quote characters. Bytes after the string are padded with binary 0 to full 32-byte length. For the version field specified by assembly language source code, see Section 7.11.7.

Table 7-169. E810 Configuration Segment [continued]

Field		Size (Bytes)	Description
Device ID Table	Device ID Table Count	4	Number of entries in the Device ID Table. May be zero.
	Device ID Table Entry 0	8	Device ID Table Entry consisting of two 4-byte fields. **Bytes 1-0:** Device ID PCI Device ID number of a device for which this configuration data applies. **Bytes 3-2:** Vendor ID PCI Vendor ID number of a device for which this configuration data applies. **Bytes 5-4:** Sub-Device ID PCI Sub-Device ID number of a device for which this configuration data applies. **Bytes 7-6:** Sub-Vendor ID PCI Sub-Vendor ID number of a device for which this configuration data applies.
	...	Nx4	All subsequent Device ID Table entries
NVM Version Table	NVM Version Count	4	Number of entries in the NVM version table. Maybe 0 if the table contains no entires.
	NVM Version Entry 0	4	NVM Version information. Details TBD.
	...		All subsequent NVM Version Table entires.
Buffers	Buffer Count	4	Number of 4KB buffers in this segment.
	Buffer 0	4096	First buffer in the segment
	...		All subsequent 4 KB buffers.

7.11.5.4 Package Buffers

A package contains one or more fixed-length Package Buffers. A Package Buffer contains a header, followed by a variable number of configuration data sections, followed by pad bytes for a total fixed size of 4096 bytes. As shown in Figure 7-34, a Package Buffer begins with a Package Buffer Header. The Package Buffer Header contains an *Index Table Size* field, a *Data End* field, and a variable length Index Table. Following the Index Table, the data for each section contained in the Package Buffer.

Table 7-170 shows the format of the Package Buffer Header.

Table 7-170. Package Buffer Header

Field	Size (Bytes)	Offset	Description
Index Table Size	2	0	Number of entries in the Index Table (1-511). A value of zero and values greater than 511 are reserved.
Data End	2	2	Byte offset from the start of the Package Buffer to the first unused byte in the buffer (12-4095). The minimum valid value of this field is 12, which accommodates the 4-byte Buffer Header and one 8-byte Index Table entry. If the entire buffer is used, this field must be set to 4096. Values less than 12 or greater than 4096 are reserved.

Table 7-170. Package Buffer Header [continued]

Field		Size (Bytes)	Offset	Description
Entry 0	Section Type	4	4	Section Type for the first section in the Package Buffer. Valid Section Type values are shown in Table 7-171.
	Section Offset	2	8	Byte offset from the start of the Package Buffer to the start of the first section in the Package Buffer (12-4095). Values less than 12 and greater than 4095 are reserved. The section offset for each entry in the index table must be aligned on a 32-bit boundary. All sections whose size is not a multiple of four bytes must be followed by pad bytes of 0xFF to the next 4-byte multiple.
	Section Size	2	10	Size in bytes of this section (1-4084). Values of zero and greater than 4084 are reserved. The section size must exclude any pad bytes added to acquire 32-bit alignment.
...	...	...	...	Subsequent Index Table entries, if any.

A Package Buffer can contain one or more Data Sections and these sections can appear in any order in the buffer. The format of a Data Section depends on the nature and purpose of the contained configuration data. The maximum possible size of a Data Section is 4084 bytes, which can occur when a Package Buffer contains only one section.

7.11.6 Section Type Enumeration

As described in Section 7.11.5, the Index Table field of the E810 Configuration Segment identifies the content, location and size of individual configuration sections. To unambiguously identify section content, the Section Type field provides a unique identifier for all possible content types.

Table 7-171 shows the Section Type values.

Table 7-171. E810 Configuration Segment Section Type Numbers

Section Type Number		Ownership Metadata Section Type Number (Section Type + 0x200)	Description	Section Reference
Dec.	Hex			
1	0x1	N/A	Segment Metadata	7.11.7
2	0x2	N/A	Security Manifest Header	7.11.7
3	0x3	N/A	Security Manifest	7.11.7
4-9	0x4-9		Reserved	
10	0xA	None	XLT0 Table for Switch	7.11.12.2.2
11	0xB	None	XLT Key Builder Table for Switch	7.11.12.2.4
12	0xC	0x20C	XLT1 Table for Switch	7.11.12.2.5
13	0xD	0x20D	XLT2 Table for the Switch	7.11.12.2.6
14	0xE	0x20E	Profile ID TCAM for the Switch	7.11.12.2.7
15	0xF	0x20F	Profile ID Redirection Table for the Switch	7.11.12.2.8
16	0x10	0x210	Field Vector Table for the Switch	7.11.12.3.1
17	0x11	None	CDID Key Builder Table for the Switch	7.11.12.2.1
18	0x12	None	CDID Redirection Table for the Switch	7.11.12.2.3

Table 7-171. E810 Configuration Segment Section Type Numbers [continued]

Section Type Number		Ownership Metadata Section Type Number (Section Type + 0x200)	Description	Section Reference
Dec.	Hex			
19	0x13		Reserved	
20	0x14	None	XLT0 Table for ACL	7.11.12.2.2
21	0x15	None	XLT Key Builder Table for ACL	7.11.12.2.4
22	0x16	0x216	XLT1 Table for ACL	7.11.12.2.5
23	0x17	0x217	XLT2 Table for ACL	7.11.12.2.6
24	0x18	0x218	Profile ID TCAM for ACL	7.11.12.2.7
25	0x19	0x219	Profile ID Redirection Table for ACL	7.11.12.2.8
26	0x1A	0x21A	Field Vector Table for ACL	7.11.12.3.1
27	0x1B	None	CDID Key Builder Table for ACL	7.11.12.2.1
28	0x1C	None	CDID Redirection Table for ACL	7.11.12.2.3
29	0x1D		Reserved	
30	0x1E	None	XLT0 Table for Flow Director	7.11.12.2.2
31	0x1F	None	XLT Key Builder Table for Flow Director	7.11.12.2.4
32	0x20	0x220	XLT1 Table for Flow Director	7.11.12.2.5
33	0x21	0x221	XLT2 Table for Flow Director	7.11.12.2.6
34	0x22	0x222	Profile ID TCAM for Flow Director	7.11.12.2.7
35	0x23	0x223	Profile ID Redirection Table for Flow Director	7.11.12.2.8
36	0x24	0x224	Field Vector Table for Flow Director	7.11.12.3.1
37	0x25	None	CDID Key Builder Table for Flow Director	7.11.12.2.1
38	0x26	None	CDID Redirection Table for Flow Director	7.11.12.2.3
39	0x27		Reserved	
40	0x28	None	XLT0 Table for RSS	7.11.12.2.2
41	0x29	None	XLT Key Builder Table for RSS	7.11.12.2.4
42	0x2A	0x22A	XLT1 Table for RSS	7.11.12.2.5
43	0x2B	0x22B	XLT2 Table for RSS	7.11.12.2.6
44	0x2C	0x22C	Profile ID TCAM for RSS	7.11.12.2.7
45	0x2D	0x22D	Profile ID Redirection Table for RSS	7.11.12.2.8
46	0x2E	0x22E	Field Vector Table for RSS	7.11.12.3.1
47	0x2F	None	CDID Key Builder Table for RSS	7.11.12.2.1
48	0x30	None	CDID Redirection Table for RSS	7.11.12.2.3
49	0x31		Reserved	
50	0x32	None	Parse Graph CAM for the Rx Parser	7.11.13.2.4
51	0x33	None	Parse Graph No-Match CAM for the Rx Parser	7.11.13.2.5
52	0x34	None	IMEM for the Rx Parser	7.11.13.7
53	0x35	None	XLT0 Key Builder Table for the Rx Parser	7.11.13.11

Table 7-171. E810 Configuration Segment Section Type Numbers [continued]

Section Type Number		Ownership Metadata Section Type Number (Section Type + 0x200)	Description	Section Reference
Dec.	Hex			
54	0x36	None	Node PType table for the Rx Parser	7.11.13.5
55	0x37	None	Marker PType TCAM for the Rx Parser	7.11.13.6
56	0x38	0x238	Boost TCAM for the Rx Parser	7.11.13.8
57	0x39	None	Protocol Group table for the Rx Parser	7.11.13.9
58	0x3A	None	Metadata Initialization table for the Rx Parser	7.11.13.14
59	0x3B	None	XLT0 Table for the Rx Parser	7.11.13.12
60	0x3C	None	Parse Graph CAM for the Tx Parser	7.11.13.2.4
61	0x3D	None	Parse Graph No-Match CAM for the Tx Parser	7.11.13.2.5
62	0x3E	None	IMEM for the Tx Parser	7.11.13.7
63	0x3F	None	XLT0 Key Builder Table for the Tx Parser	7.11.13.11
64	0x40	None	Node PType table for the Tx Parser.	7.11.13.5
65	0x41	None	Marker PType TCAM for the Tx Parser.	7.11.13.6
66	0x42	0x242	Boost TCAM for the Tx Parser.	7.11.13.8
67	0x43	None	Protocol Group table for the Tx Parser	7.11.13.9
68	0x44	None	Metadata Initialization table for the Tx Parser	7.11.13.14
69	0x45	None	XLT0 Table for the Tx Parser	7.11.13.12
70	0x46	None	Initialization ID Redirection Table for the Rx Parser	7.11.13.13
71	0x47	None	Initialization ID Redirection Table for the Tx Parser	7.11.13.13
72	0x48	None	Marker Group ID Table for the Rx Parser	7.11.13.10
73	0x49	None	Marker Group ID Table for the Tx Parser	7.11.13.10
74	0x4A	None	Last Protocol Table for the Rx Parser	7.11.13.15
75	0x4B	None	Last Protocol Table for the Tx Parser	7.11.13.15
76	0x4C	None	PG Spill CAM for the Rx Parser	7.11.13.3
77	0x4D	None	PG Spill CAM for the Tx Parser	7.11.13.3
78	0x4E	None	No-Match Spill CAM for the Rx Parser	7.11.13.4
79	0x4F	None	No-Match Spill CAM for the Tx Parser	7.11.13.4
80	0x50	None	XLT0 Table for the Protocol Engine[1]	7.11.12.2.2
81	0x51	None	XLT Key Builder Table for the Protocol Engine	7.11.12.2.4
82	0x52	0x252	XLT1 Table for the Protocol Engine	7.11.12.2.5
83	0x53	0x253	XLT2 Table for the Protocol Engine	7.11.12.2.6
84	0x54	0x254	Profile ID TCAM for the Protocol Engine	7.11.12.2.7
85	0x55	0x255	Profile ID Redirection Table for the Protocol Engine	7.11.12.2.8
86	0x56	0x256	Field Vector Table for the Protocol Engine	7.11.12.3.1
87	0x57	None	CDID Key Builder Table for Protocol Engine	7.11.12.2.1
88	0x58	None	CDID Redirection Table for Protocol Engine	7.11.12.2.3

Table 7-171. E810 Configuration Segment Section Type Numbers [continued]

Section Type Number		Ownership Metadata Section Type Number (Section Type + 0x200)	Description	Section Reference
Dec.	Hex			
89	0x59		Reserved	
90	0x5A	None	Miscellaneous configuration for Rx Parser	7.11.13.16
91	0x5B	None	Miscellaneous configuration for Tx Parser	7.11.13.16
92	0x5C	None	Recipe configuration for the Switch	7.11.12.5
93	0x5D	None	Recipe-to-Profile associations for the Switch	7.11.12.6
94	0x5E	None	Rx Flex Descriptors for the HIF	7.11.14.2
95	0x5F	None	PTYPE Translation Table for the RDPU	7.11.15.2
96	0x60	None	PROTOCOL Table for RDPU	7.11.15.3
97	0x61	None	Flags Redirection Table for the Rx Parser	7.11.13.17
98	0x62	None	Flags Redirection Table for the Tx Parser	7.11.13.17
99	0x63	None	Field Vector Table for Manageability	7.11.12.3.1
100	0x64	None	Miscellaneous Profile and Control Domain Configuration for the Switch	7.11.12.6.1
101	0x65	None	Miscellaneous Profile and Control Domain Configuration for ACL	7.11.12.6.1
102	0x66	None	Miscellaneous Profile and Control Domain Configuration for Flow Director	7.11.12.6.1
103	0x67	None	Miscellaneous Profile and Control Domain Configuration for RSS	7.11.12.6.1
104	0x68	None	Miscellaneous Profile and Control Domain Configuration for the Protocol Engine	7.11.12.6.1
105	0x69	None	Mask Select Filters for RSS	7.11.12.6.2
106	0x6A	None	Mask Select Filters for Flow Director	7.11.12.6.2
107	0x6B	None	Mask Select Filters for the Protocol Engine	7.11.12.6.2
108	0x6C	None	Quad Hash Control Table for the Protocol Engine	7.11.12.6.3
All other values			Reserved	

1. Documents might also refer to the PE as the Quad-Hash (QH) block.

Table 7-172. Section Details and Sizes

Section	Maximum Number of Instances	Fixed Header Size (Bytes)	Entry Size (Bytes)	Number of Entries	Section Size (Bytes)	Owner Section Size (Bytes)[1]	Maximum Total (Bytes)
1	1	4	32	1	36	0	36
CDID Key Builder	5	0	17	1	17	0	85
XLT0	5	4	1	2048	2052	0	10260
CDID Redirection Table	5	0	1	7	7	0	35
XLT Key Builder	5	12	24	8	204	0	1020
XLT1	5	4	1	8192	8196	8200	81980
XLT2	5	4	2	1024	2052	1032	15420
Profile ID TCAM (SW)	1	2	13	512	6658	520	7178

Table 7-172. Section Details and Sizes [continued]

Section	Maximum Number of Instances	Fixed Header Size (Bytes)	Entry Size (Bytes)	Number of Entries	Section Size (Bytes)	Owner Section Size (Bytes)[1]	Maximum Total (Bytes)
Profile ID TCAM (ACL)	1	2	13	512	6658	520	7178
Profile ID TCAM (PE)	1	2	13	64	834	72	906
Profile ID TCAM (RSS)	1	2	13	512	6658	520	7178
Profile ID TCAM (FD)	1	2	13	512	6658	520	7178
Profile ID Redirection (SW)	1	4	1	256	260	264	524
Profile ID Redirection (ACL)	1	4	1	128	132	136	268
Profile ID Redirection (PE)	1	4	1	32	36	40	76
Profile ID Redirection (RSS)	1	4	1	128	132	136	268
Profile ID Redirection (FD)	1	4	1	128	132	136	268
FV Extraction (SW)	1	4	192	256	24580	264	49420
FV Extraction (ACL)	1	4	128	128	8196	136	16524
FV Extraction (PE)	1	4	96	32	1540	40	3116
FV Extraction (RSS)	1	4	96	128	6148	136	12428
FV Extraction (FD)	1	4	96	128	6148	136	12428
FV Extraction (MNG)	1	4	128	132	132	0	132
PG CAM	2	4	16	2048	32772	0	65544
No-Match CAM	2	4	12	1024	12292	0	24584
Marker PType TCAM	2	4	24	1024	24580	0	49160
IMEM	2	4	48	192	9220	0	18440
Boost TCAM	2	4	88	256	22532	264	45592
Protocol Group Table	2	4	24	192	4612	0	9224
Marker Group Table	2	4	8	128	1028	0	2056
Parser XLT0 Key Builder Table	2	0	4	1	4	0	8
Parser XLT0 Table	2	4	1	1024	1028	0	2056
Parser Initialization ID Redir	2	0	1	14	14	0	28
Parser Metadata Init Table	2	4	24	16	388	0	776
Last Protocol Table	2	0	4	6	24	0	48
PG Spill CAM	2	4	17	128	2180	0	4360
PG No-Match Spill CAM	2	4	13	64	836	0	1672
Parser Misc. Configuration	2	0	24	1	24	0	48
Switch Recipes	1	4	40	64	2564	0	2564
Profile/Switch Association	1	4	8	256	2052	0	2052
Rx Flex Descriptor	1	4	44	61	2688	0	2688
PTYPE_REMAP	1	4	1	1024	1028	0	1028
PROTOCOL for RDPU	1	4	28	256	7172	0	7172

Table 7-172. Section Details and Sizes [continued]

Section	Maximum Number of Instances	Fixed Header Size (Bytes)	Entry Size (Bytes)	Number of Entries	Section Size (Bytes)	Owner Section Size (Bytes)[1]	Maximum Total (Bytes)
Flags Redirection	2	4	1	64	68	0	136
						Total:	**473142**

1. Owner section size of 0 indicates that the section cannot be modified with an Update Package command.

7.11.7 Segment Metadata Section

The E810 Configuration Segment Metadata Section contains user defined metadata about the segment. This section has a Section Type value of 1. The Get Package Info List command (see Section 7.11.9.4) returns the information contained in this section.

The structure of this section is shown in Table 7-173.

Table 7-173. Segment Metadata Section

Field	Size (Bytes)	Description
Segment Version	4	The version of the segment. The version number follows the convention defined in Section 7.11.4 Software tools set this value from the version statement in an assembly language source file. Software tools can reserve the value 255.255.255.255 to mean the version is unknown.
Segment Name	32	Free form UTF-8 null-terminated 28-byte name of the segment. Software tools set this value from the name statement in an assembly language source file. Software tools can reserve a zero length string (first byte is binary 0) to mean the name is unknown. Software tools enforce that at least the 28th byte of the name must be a binary 0.
Track ID	4	The Track ID is a numerical representation of the package type and uniquely identifies the type of package as specified by a track statement at segment scope in an assembly source file. The Track ID and package name are immutable for each package type; OS, Comms, etc. Software tools can use 0xFFFFFFFF as a reserved value meaning the Track ID is unknown.

7.11.8 Segment Security Manifest

This section defines the structure of the ICE Segment Security Manifest. This manifest enables firmware or other software to verify selected segment buffers are genuine before transferring configuration to control registers.

7.11.8.1 Security Manifest Overview

The segment security manifest consists of two parts.

- An ordered list of secure hashes, one for each 4 KB buffer in the remainder of the segment.
- The PKCS v2.2 structure returned from the signing server. This structure contains a signature, a manifest hash of the buffer hashes and the security version number.

The chain of trust for downloaded package buffers is as follows:

- Firmware verifies the manifest hash using the Intel RSA signature.
- Firmware verifies the list of buffer hashes using the manifest hash.

- Firmware verifies each buffer using corresponding buffer hash.

The error messages that are issued when firmware detects an issue in the security manifest are as follows:

- **Missing security manifest** — Return missing signature error on AQ command. Firmware should not issue CORER.
- **RSA signature mismatch on the manifest hash** - Return bad signature error on AQ command. Firmware should not issue CORER.
- S**ecure Version Number too low** — Return bad SVN error on AQ command. Firmware should not issue CORER.
- **Manifest hash mismatch on the buffer hashes** — Return bad manifest error on AQ command. Firmware must issue CORER.
- **Buffer hash mismatch on the buffer** — Return bad buffer hash error on AQ command. Firmware must issue CORER.
- **Non-security error in buffer processing** — Return appropriate error on AQ command. Firmware must issue CORER.

7.11.8.2 Protection Provided by a Segment Security Manifest

A segment security manifest provides a limited scope of protection within a package file. Specifically, the security manifest protects only parts of the ICE segment, not the entire segment nor package. The parts of a package not covered by a security segment rely on host security for protection, for example, file system root access. The rationale is that a corrupted host cannot be trusted to check itself, therefore a security manifest can protect only the buffers visible to firmware. See Table 7-174 for an overview.

Table 7-174. Content Protected or Excluded by a Segment Security Manifest

Package Content Type	Protected?
Package Header	No
Package Notes Segment	No
Package Global Metadata Segment	No
Segment Header	No
Other device configuration segments	No
Segment buffers transferred to firmware with Download Package AQ command	Yes
Segment buffers NOT transferred to firmware. (or example, segment metadata)	No

7.11.8.3 Segment Security Requirements

Production packages downloaded by the host via the Download Package AQ command must have a valid security manifest in the ICE configuration segment. Non-production packages might have a valid or invalid security manifest.

7.11.8.3.1 Development Mode

To facilitate package development, firmware must support a permissive mode that bypasses security manifest checking. In this mode, the downloaded package might omit the security manifest. If the package contains a security manifest, firmware must ignore the manifest content.

Table 7-175 shows the security enforcement options. These options depend on the CSR protection fuse and the NVM *Security Enable* bit.

Table 7-175. Package Manifest Enforcement Options

CSR Protection Enable (GL_UFUSE_SOC[6])	NVM_SEC_EN (NVM Offset 0x02)	Firmware Requires Valid Package Security Manifest
0	0	No
0	1	Yes
1	0	Yes
1	1	Yes

7.11.8.3.2 NVM Packages

The ICE NVM signature already provides protection for the packages stored in NVM. To conserve NVM storage, these package should omit the security manifest. If an NVM package does contain a security manifest, firmware must ignore the manifest content.

7.11.8.4 Security Manifest Section Numbers

Table 7-176 provides section numbers associated with the security manifest:

Table 7-176. Security Manifest Section Numbers

Section	Number
Security Manifest Header	2 (0x2)
Security Manifest	3 (0x3)

7.11.8.5 Security Manifest Header Section

The manifest header section consists of the PKCS v2.2 structure.

Table 7-177. Security Manifest Header Section Fields

Field	Size (Bits)	CSR	Description
Module Type	32	N/A	The module type. Must be 0x6.
Header Length	32	N/A	The header length. Must be 0xA1.
Header Version	32	N/A	The header version. Must be 0x000100000.
Module ID	32	N/A	The module ID. Must be 0x0.
Module Vendor	32	N/A	The vendor ID. Must be 0x8086.
Date	32	N/A	The date on which the signature was created.
Size	32	N/A	The number of 32-bit (DWORD) units. Must be 0xC8.

Table 7-177. Security Manifest Header Section Fields [continued]

Field	Size (Bits)	CSR	Description
Key Size	32	N/A	The size of the key. Must be 0x40.
Modulus Size	32	N/A	The modulus size. Must be 0x40.
Exponent Size	32	N/A	The size of the exponent. Must be 0x1.
SVN	32	N/A	Secure version number. This monotonically increasing number represents the security version of this package. If this number is less than the persistent SVN number stored in the NVM, then firmware will reject this package. Otherwise, firmware accepts the package.
Reserved	32	N/A	Reserved (0).
Reserved	32	N/A	Reserved (0).
Reserved	32	N/A	Reserved (0).
Unique ID	32	N/A	Unique ID of this configuration, taken from the support team ETrack ID.
Segment Type	16	N/A	The segment type. 0x10 for the ICE configuration segment.
Reserved	16	N/A	Reserved (0).
Reserved	512	N/A	Reserved (0).
Public Key Modulus	2048	N/A	The public key. The signing server generates the content of this field.
RSA Exponent	32	N/A	RSA Exponent used for the signature calculation. The signing server generates this field.
Signature	2048	N/A	The signature created by the signing server. The signature served must be passed the security manifest section header section only for signing.
Reserved	128	N/A	Reserved area used by the signing server.
Manifest Hash	256	N/A	The SHA-256 of the two buffers containing the security sections.
Reserved	864	N/A	Reserved (0).

7.11.8.6 Security Manifest Section

A manifest section contains SHA256 hash values corresponding to each buffer downloaded to firmware in the segment. Package build tools must place each hash value in the manifest section in the same order as the corresponding configuration buffer. A buffer containing a manifest section must not contain any other type of section.

Table 7-178 shows the format of a manifest section:

Table 7-178. Security Manifest Section Fields

Field	Size (Bits)	CSR	Description
Count	16	N/A	Number of SHA256 hashes in this section. Valid values are 0-127.
Offset	16	N/A	Offset to the first hash in this section.
Hash0	256	N/A	SHA256 hash corresponding to a 4 KB download package buffer.
...	Nx256	N/A	All other SHA256 hashes in this section.

7.11.8.7 Security Manifest Section Location in the ICE Segment

Package build tools must place security manifest sections before any other section downloaded to firmware. Specifically section 2 must be the first section, followed by exactly two section 3's. These sections occupy the first 4 KB buffers downloaded to firmware. In the package, build tools can place sections not downloaded to firmware before security manifest sections.

7.11.9 Package Configuration Admin Commands

7.11.9.1 Download Package Command (0x0C40)

This section specifies the Download Package AQ command. This command configures the packet processing pipeline, overwriting the previous configuration. During initialization, the host can issue multiple Download Package commands, each containing one Package Buffer, until package download completes. The host sets the *Last Buffer* Flag in the command descriptor for the last Download Package command of the sequence. The host must hold the Global Configuration Lock resource using the Request Resource Ownership Admin command (Section 9.5.13.6) to issue this command.

Table 7-179. Download Package Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0C40	Command opcode.
Data Length	4-5	28-4096	Length of command/response buffer. Varies depending on package content.
Return Value/VFID	6-7		Return value. Cleared by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion descriptor for this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion descriptor for this command.
Flags	16		**Bit 0:** Last Buffer When set, this Download Package command is the last in the sequence. All other bits reserved.
Reserved	17-23		Reserved.
Data Address High	24-27		Host memory address of the command buffer.
Data Address Low	28-31		

Table 7-180. Download Package Command Buffer Format

Field	Size (Bytes)	Description
Package	12-4096	Package Buffer as defined in Section 7.11.5.4.

Table 7-181. Download Package Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0C40	Command opcode.
Data Length	4-5	28-4096	Length of command/response buffer. Varies depending on package content.
Return Value/VFID	6-7		Return code written by firmware. In the case of a failure, the existing configuration is unmodified. SUCCESS = Package download succeeded. EFAULT = *Data Address* field was 0. ENOMEM = Insufficient hardware resources exist to perform this configuration. EINVAL = An element within the package data was invalid. *Error Offset* contains the byte offset into the package data of the invalid element. *Error Info* contains an additional error code describing the problem with the element. EACCES = Attempt to overwrite the default package. ENOSEC = Missing security manifest. EBADSIG = Bad RSA signature. ESVN = SVN number prohibits usage of this package. EBADMAN = Manifest hash mismatch. EBADBUF = Buffer hash mismatches expected value in manifest.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion descriptor for this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion descriptor for this command.
Error Offset	16-19		Offset to an invalid element detected by firmware in the downloaded package. Firmware sets this field to zero on success.
Error Info	20-23		Contains a code for additional error information that describes the nature of the problem found, if any. Firmware sets this field to zero on success.
Data Address High	24-27		Host memory address of the command buffer.
Data Address Low	28-31		

7.11.9.2 Upload Section Command (0x0C41)

This section specifies the Upload Section AQ command. This command retrieves hardware configuration for requested sections. The host can optionally acquire the Change Lock before issuing this command. Holding the Change Lock is not required, but guarantees that returned section data is not a partially modified state due to Update Package commands from another PF. This command is read-only and does not modify the current hardware configuration.

Table 7-182. Upload Section Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0C41	Command opcode.
Data Length	4-5	28-4096	Length of command/response buffer. Varies depending on package content.
Return Value/VFID	6-7		Return value. Cleared by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion descriptor for this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion descriptor for this command.
Reserved	16-23		Reserved.
Data Address High	24-27		Host memory address of the command buffer.
Data Address Low	28-31		

Table 7-183. Upload Section Command Buffer Format

Field	Size (Bytes)	Description
Package	12-4096	Package Buffer as defined in Section 7.11.5.4.

Table 7-184. Upload Section Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0C41	Command opcode.
Data Length	4-5	28-4096	Length of command/response buffer. Varies depending on package content.
Return Value/VFID	6-7		Return code written by firmware. SUCCESS = Section upload succeeded. EFAULT = *Data Address* field was 0. EINVAL = Unrecognized Section Number. ERANGE = Invalid Section Offset.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion descriptor for this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion descriptor for this command.
Reserved	16-23		Reserved.
Data Address High	24-27		Host memory address of the command buffer.
Data Address Low	28-31		

7.11.9.3 Update Package Command (0x0C42)

This section specifies the Update Package AQ command. This command modifies the configuration of the packet processing pipeline without erasing the existing configuration. The host can issue multiple Update Package commands, containing one Package Buffer each, until pipeline configuration update is complete. The host must hold the Global Configuration Lock or the Change Lock resources using the Request Resource Ownership Admin command (Section 9.5.13.6) to issue this command.

The AQ command format for this command is almost identical to the Download Package command (Section 7.11.9.1).

Table 7-185. Update Package Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0C42	Command opcode.
Data Length	4-5	28-4096	Length of command/response buffer. Varies depending on package content.
Return Value/VFID	6-7		Return value. Cleared by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion descriptor for this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion descriptor for this command.
Flags	16		**Bit 0:** Last Buffer When set, this Update Package command is the last in the sequence. All other bits reserved.
Reserved	17-23		Reserved.
Data Address High	24-27		Host memory address of the command buffer.
Data Address Low	28-31		

Table 7-186. Update Package Command Buffer Format

Field	Size (Bytes)	Description
Package	12-4096	Package Buffer as defined in Section 7.11.5.4.

Table 7-187. Update Package Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0C42	Command opcode.
Data Length	4-5	28-4096	Length of command/response buffer. Varies depending on package content.
Return Value/VFID	6-7		Return code written by firmware. In the case of a failure, the existing configuration is unmodified. SUCCESS = Package download succeeded. EFAULT = *Data Address* field was 0. ENOMEM = Insufficient hardware resources exist to perform this configuration. EINVAL = An element within the package data was invalid. *Error Offset* contains the byte offset into the package data of the invalid element. *Error Info* contains an additional error code describing the problem with the element. EACCES = Attempt to overwrite the default package.

Table 7-187. Update Package Response [continued]

Name	Byte.Bit	Value	Remarks
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion descriptor for this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion descriptor for this command.
Error Offset	16-19		Offset to an invalid element detected by firmware in the downloaded package. Firmware sets this field to zero on success.
Error Info	20-23		Contains a code for additional error information that describes the nature of the problem found, if any. Firmware sets this field to zero on success.
Data Address High	24-27		Host memory address of the command buffer.
Data Address Low	28-31		

7.11.9.4 Get Package Info List Command (0x0C43)

This command populates a host buffer with information about the currently installed packages. The package information structure is defined in the Response Buffer table. This command blocks until no other driver holds the Change Lock nor the Global Configuration Lock. This command does not block when called by the driver currently holding the Change or Global Configuration lock. Firmware can return zero or more Package Info structures in the response buffer. Firmware returns zero Package Info structures when NVM does not contain any packages.

The information returned in by this command corresponds to the content of Section 1 within the E810 Configuration Segment (see Section 7.11.7).

Table 7-188. Get Package Info List Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0C43	Command opcode.
Data Length	4-5	4096	Length of command/response buffer.
Return Value/VFID	6-7		Return value. Cleared by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion descriptor for this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion descriptor for this command.
Reserved	16-23		Reserved.
Data Address High	24-27		Host memory address of the command buffer.
Data Address Low	28-31		

Table 7-189. Get Package Info List Command Response Buffer Format

Field	Size (Bytes)	Description
Count	4	Number of Package Info structures returned in this buffer.
Package Info Array	Variable	Packed array of Package Info structures. Table 7-190 shows the format of a Package Info structure.

Table 7-190. Package Info Structure

Field	Size (Bytes)	Description
Version	4	M.m.u.d version of this package as specified in Section 7.11.4.
Name	32	Free form (null terminated UTF-8 by convention) 32-byte name of the package. The 32nd byte of the name is always a binary 0. The Intel factory default package will have the name 'DEFAULT' padded with zeros to 32-byte length.
Is In NVM	1	1 if this package is stored in NVM. 0 otherwise.
Is Active	1	1 if this package is currently active (that is, able to process packets). 0 otherwise.
Is Active At Boot	1	1 if firmware enables this package automatically at boot time. 0 otherwise. This flag is always 0 if Is In NVM is 0.
Is Modified	1	1 if the host modified this package with an Update Package command. 0 otherwise.

7.11.10 Uniform TCAM Key Encoding

Some sections define the content of Ternary Content Addressable Memories (TCAMs). In addition to matching an input of 0 or 1, TCAM keys also define don't care and invalid input conditions. All TCAM keys use a key/key-invert encoding to represent the four input conditions, as show in Table 7-81, "Key and Key Invert Programming Effect".

The key is arranged as a sequence of key and key invert, as follows:

(key_invert << key.size()) | (key)

Tools must enforce that only a single invalid input bit (~) can be used in a TCAM key.

The TCAM keys must be encoded as described in Table 7-81.

7.11.10.1 Example 1

A 3-bit input value of 'b10? is represented in the TCAM with key=011 and key invert of 101. The key matches two possible binary input values 100 and 101.

7.11.10.2 Example 2

A 3-bit input value of 'b10~ is represented in the TCAM with key=010 and key invert=100. The key matches no input values due to the never-match condition in bit 0.

7.11.11 Ownership Configuration

The package can explicitly designate ownership of specific hardware resources. Firmware can use this ownership information to enforce correct runtime behavior. For example, firmware can prevent PF3 from issuing an Update Package command that alters a hardware resource belonging to PF1.

Ownership information for a package is contained in dedicated ownership sections. The host can download ownership sections only in a Download Package command. Ownership sections are not allowed in an Update Package command. Therefore, resource ownership is static and cannot be modified at runtime. The Host can retrieve ownership information using the Upload Section command.

The Section Type Number for ownership metadata is the Section Type Number of the resource itself plus 0x200. Table 7-171.

All ownership sections use a common format. The section number defines the specific hardware resource to which the ownership applies. Table 7-191 shows the format of an ownership section.

The package must specify ownership for any resource that cab be modified at runtime with an Update Package command (for example, for UDP tunnel configurations).

A package can omit an ownership section. In this case, the default policy to deny write-access applies to the corresponding hardware resource.

An ownership section can omit leading and trailing entries in the range of resources covered. For example, a hardware resource with 1024 entries can have an ownership section that defines only entries 100 to 200.

A package can contain multiple ownership sections for the same hardware resource. In the case of a conflict in ownership, the last section downloaded takes precedence.

Table 7-191. Package Format for Ownership Configuration

Field	Size (Bytes)	Description
Count	2	Number of entries in this section.
Reserved	2	Reserved (0).
Starting Resource Number	4	A resource number appropriate for the corresponding hardware. The exact meaning of this number depends on the section number: CAM, TCAMs, BICAMS, TABLES: Resource number is the address. For example, ownership for a TCAM with 192 entries will have Resource Numbers 0-191. IMEM: Resource number is the IMEM address
Owner 0	1	Owner for this resource. Owners values are defined as follows: 0-7 = Write access allowed for a specific PF 0-7 8 = Global 9 = Unmanaged 10 = Shared (writable by any PF after allocation) 11-254 = Reserved 255 = Unspecified (access denied to any PF)
All other Entries	Nx1	Subsequent consecutive owner values. The Resource Number for each consecutive owner is the previous Resource Number plus one.

The specific behavior of each allocation type is shown in Table 7-192.

Table 7-192. Resource Ownership Types and Behavior

	PF# (0-7)	Global (8)	Unmanaged (9)	Shared[1] (10)	Unspecified
Description	Dedicated to a specific PF	Read-only resource available globally	The resource ownership is not managed by the firmware.	Any PF can allocate as shared or private	Package did not specify an owner
PF Allocate	No	No	No	Yes, by each owner	No
PF Release	No	No	No	Yes, last release returns resource to a free pool	No
PF Read Access	By owner	Yes by any PF	By any owner	By any owner	No

Table 7-192. Resource Ownership Types and Behavior [continued]

	PF# (0-7)	Global (8)	Unmanaged (9)	Shared[1] (10)	Unspecified
PF Write Access	By owner	No	By any owner	By any owner	No
Firmware Tracking	Static	Static	No tracking	Dynamic tracks all owners	None
Firmware Cleanup	PFR, CORER	CORER	CORER	On last release, PFR - PF related settings only, CORER	None

1. A shared owner can be assigned only for Profile ID TCAM (sections 0x20E, 0x218, 0x222, 0x22C, 0x254) and the Field Vector Table (sections 0x210, 0x21A, 0x224, 0x22E 0x256).

7.11.12 Common Packet Profile Commands

7.11.12.1 Introduction

Packets proceed down the pipeline accompanied by a variety of metadata, such as the packet type produced by the parser. Pipeline blocks use this metadata to help determine how to process the packet.

As-is, the metadata can be inconvenient or insufficient to directly control block behavior. For proper processing, each pipeline block contains a mechanism to map metadata to block specific Profile IDs. Each block uses its own unique Profile ID enumeration to precisely control packet processing.

Each block defines its own unique Profile IDs, but all blocks share a common methodology for this configuration. As described in this section, the E810 uses a common set of commands to define the Profile ID mappings for each block.

Configuration of Profile IDs can occur in three ways:

- As part of the initialization package loaded by firmware from NVM.
- As part of an initialization package provided by the host via the Download Package AQ command.
- Runtime configuration by the host.

7.11.12.2 Package Buffer Formats

This section specifies the format of package buffers related to Profile ID configuration.

7.11.12.2.1 Package Format for CDID Key Builder Tables

This section specifies the package format of the per-block CDID Key Builder Tables. These tables select metadata fields used as input in to the Control Domain ID table (XLT0).

Table 7-193 shows the section type enumeration.

Table 7-193. Package Section Type Numbers for CDID Key Builder Tables

Section Type Number	Allowed in Update Package?	Description
17	No	CDID Key Builder Table for the Switch.
27	No	CDID Key Builder Table for the ACL.

Table 7-193. Package Section Type Numbers for CDID Key Builder Tables [continued]

Section Type Number	Allowed in Update Package?	Description
37	No	CDID Key Builder Table for the Flow Director.
47	No	CDID Key Builder Table for RSS.
87	No	CDID Key Builder Table for PE.

Table 7-194 shows the format of the CDID Key Builder Table for each block.

Table 7-194. Package Buffer Format for the CDID Key Builder Table

Field	Size (Bytes)	Description
Direction Select Flag 0	2	Index of the first flag used to define the direction. Valid values are 0-511. Value of 0xFFFF is reserved for internal use by assembler tools before conversion to a valid value. All other values are reserved. The value in this field corresponds to an individual bit from any metadata field with metadata ID in range 0-31. For example: Direction Select 0-15 selects bit 0-15 of Metadata ID 0. Direction Select 16-31 selects bit 0-15 of Metadata ID 1. Direction Select 32-47 selects bit 0-15 of Metadata ID 2. and so on up to Metadata ID 31. Relevant CSR: GL_<block>_FLGS_L1SEL0_1[8:0] When no direction flags are used, software tools write a zero to this field.
Direction Select Flag 1	2	Index of the second flag used to define the direction. Valid values are 0-511. Value of 0xFFFF is reserved for internal use by assembler tools before conversion to a valid value. All other values are reserved. Values are defined in same manor as Direction Select Flag 0. Relevant CSR: GL_<block>_FLGS_L1SEL0_1[24:16] When no direction flags are used, software tools write a zero to this field.
Direction Select Flag 2	2	Index of the third flag used to define the direction. Valid values are 0-511. Value of 0xFFFF is reserved for internal use by assembler tools before conversion to a valid value. All other values are reserved. Values are defined in same manor as Direction Select Flag 0. Relevant CSR: GL_<block>_FLGS_L1SEL2_3[8:0] When no direction flags are used, software tools write a zero to this field.
Direction Select Flag 3	2	Index of the fourth flag used to define the direction. Valid values are 0-511. Value of 0xFFFF is reserved for internal use by assembler tools before conversion to a valid value. All other values are reserved. Values are defined in same manor as Direction Select Flag 0. Relevant CSR: GL_<block>_FLGS_L1SEL2_3[24:16] When no direction flags are used, software tools write a zero to this field.
Direction Select	4	This 32-bit value creates a bit mapping that produces the Dir[1:0] direction control. This field is partitioned into two 16-bit values as follows: [15:0] = These bits provide a 4-bit to 1 mapping to produce Dir[0]. The four flag bits selected above create an index value (0-15) selecting one bit of this [15:0] range. [31:16] = These bits provide a 4-bit to 1 mapping to produce Dir[1]. The four flag bits selected above create an index value (0-15) selecting one bit of this [31:16] range. Relevant CSR: GL_<block>EXT_FLGS_L1TBL[31:0] When no direction flags are used, software tools write a zero to this field.

Table 7-194. Package Buffer Format for the CDID Key Builder Table [continued]

Field	Size (Bytes)	Description
Metadata Rx Select (DIR=0)	1	This field specifies the Metadata ID value used to select the MD0 input into XLT0 for Receive (Rx) traffic or for all traffic when the XLT0 Partition Mode field is 0. Valid values are 0-31. Value of 0xFF is reserved for internal use by assembler tools before conversion to a valid value. All other values are reserved. Relevant CSR: GL_<block>_CDMD_L1Sel[4:0]
Metadata Tx Select (DIR=1)	1	This field specifies the Metadata ID value used to select the MD0 input into XLT0 for Transmit (Tx) traffic. Valid values are 0-31. Value of 0xFF is reserved for internal use by assembler tools before conversion to a valid value or when the XLT0 Partition Mode field is 0. All other values are reserved. Relevant CSR: GL_<block>_CDMD_L1Sel[12:8]
Metadata Aux0 Select (DIR=2)	1	This field specifies the Metadata ID value used to select the MD0 input into XLT0 for Aux0 traffic. Valid values are 0-31. Value of 0xFF is reserved for internal use by assembler tools before conversion to a valid value or when the XLT0 Partition Mode field is 0. All other values are reserved. Relevant CSR: GL_<block>_CDMD_L1Sel[20:16]
Metadata Aux1 Select (DIR=3)	1	This field specifies the Metadata ID value used to select the MD0 input into XLT0 for Aux1 traffic. Valid values are 0-31. Value of 0xFF is reserved for internal use by assembler tools before conversion to a valid value or when the XLT0 Partition Mode field is 0. All other values are reserved. Relevant CSR: GL_<block>_CDMD_L1Sel[28:24]
XLT0 Partition Mode (Bi-dir. Pipe Enable)	1	This field controls the XLT0 Table MUX selection. Valid values are 0 to 2. All other values are reserved. See Table 7-195. Relevant CSR: GL_<block>_CDMD_L1Sel[31:30]

Table 7-195 defines the XLT0 Partition Mode values.

Table 7-195. XLT0 Partition Mode Selection Values

Value	Description
0	XLT0 Table lookup value is metadata[10:0].
1	XLT0 Table lookup value is composed as Dir[0] as the most significant bit, plus the selected metadata[9:0] as the least significant bits.
2	XLT0 Table lookup value is composed as Dir[1:0] as the most significant bits, plus the selected metadata[8:0] as the least significant bits.
3-254	Reserved.
255	Reserved for internal use by assembler tools before conversion to a valid value.

7.11.12.2.2 Package format of the XLT0 Table

This section specifies the package format for configuring the per-block XLT0 table. The XLT0 Table provides an intermediate stage during calculation of a Control Domain ID.

Each pipeline block uses a distinct XLT0 Table configuration and distinct Package Section Type Numbers to contain the configuration data. Table 7-196 shows the section type enumeration.

Table 7-196. Package Section Type Numbers for XLT0 Tables

Section Type Number	Allowed in Update Package?	Description
10	No	XLT0 Table for the Switch.
20	No	XLT0 Table for the ACL.
30	No	XLT0 Table for the Flow Director.
40	No	XLT0 Table for RSS.
80	No	XLT0 Table for PE.

The package format for XLT0 Table configuration is common for all blocks as shown in Table 7-197.

Table 7-197. Package Format for XLT0 Table Configuration

Field	Size (Bytes)	Description
Count	2	Number of entries in this section. If the XLT0 table is split into multiple sections, this field describes only the number of entries contained in this section.
Offset	2	Byte offset from the start of the table to the first value in this section. The maximum effective offset for any value is 2047. Effective offsets greater than 2047 are reserved. The offset field incorporate the direction, if used, in the key value. No Direction Bits: 0-2047 = Control Domain value 1 Direction Bit: 0-1023 = Control domain for DIR=0 1024-2047 =Control domain for DIR=1 2 Direction Bits: 0-511 = Control domain value for DIR=0 512-1023 = Control domain value for DIR=1 1024-1535 =Control domain value for DIR=2 1536-2047 =Control domain value for DIR=3 Relevant CSR: GL_<block>_XLT0_L1ADDR.*LINE_IDX*. Should be set together with *AUTO_INC* bit.
Value	1	Control Domain value of this XLT0 entry. Valid values are in range (0-7). Value of 255 is reserved for internal use by assembler tools before conversion to a valid value. All other values are reserved. Relevant CSR from “Non-TCAM Tables”: GL_<block>_XLT0_L1DATA.*DATA*
...	...	Subsequent values at consecutive offsets.

7.11.12.2.3 CDID Redirection Table

After the XLT0 table generates the CDID, the CDID Redirection Table can change the CDID to a different value. The CDID Redirection Table always maps the input CDID to an output CDID. Therefore, to allow CDID values to pass through unchanged, the host must initialize this table to the trivial identity mapped case (for example, CDID 7 maps to CDID 7).

Each block in the pipeline contains its own CDID Redirection Table. Table 7-198 shows the relevant Package Section Type for each block.

Table 7-198. Package Section Type Numbers for CDID Redirection Tables

Section Type Number	Allowed in Update Package?	Description
18	No	CDID Redirection Table for the Switch.
28	No	CDID Redirection Table for the ACL.
38	No	CDID Redirection Table for the Flow Director.
48	No	CDID Redirection Table for RSS.
88	No	CDID Redirection Table for PE.

The package format for the CDID Redirection Table is common for all blocks as shown in Table 7-199.

Table 7-199. Package Format for CDID Redirection Table

Field	Size (Bits)	Description
CDID 0 Redirect Value	4	Redirected value for CDID 0. Relevant CSRs: GL_<block>_P2P_L1ADDR.*LINE_IDX* = 0 GL_<block>_P2P_L1DATA.*DATA*
CDID 1 Redirect Value	4	Redirected value for CDID 1. Relevant CSRs: GL_<block>_P2P_L1ADDR.*LINE_IDX* = 0 GL_<block>_P2P_L1DATA.*DATA*
CDID 2-15 Redirect Values	14 x 4	Redirected value for CDID 2 to 15. Relevant CSRs: GL_<block>_P2P_L1ADDR.*LINE_IDX* GL_<block>_P2P_L1DATA.*DATA*

Note: Initial value for GL_<block>_P2P_L1ADDR should be 0x1 << 31 (*AUTO_INC* field set).

7.11.12.2.4 Package Format of XLT Key Builder Tables

This section specifies the package format of per-block XLT Key Builder Tables. The XLT Key Builder Table selects the metadata and flags fields used as input into the Profile ID Compression (XLT1) and Expansion (XLT2) tables.

Each pipeline block uses a distinct XLT Key Builder Table configuration and distinct Package Section Type Numbers to contain the configuration data. Table 7-200 shows the section type enumeration.

Table 7-200. Package Section Type Numbers for XLT Key Builder Tables

Section Type Number	Allowed in Update Package?	Description
11	No	XLT Key Builder Table for the Switch.
21	No	XLT Key Builder Table for the ACL.
31	No	XLT Key Builder Table for the Flow Director.
41	No	XLT Key Builder Table for RSS.
81	No	XLT Key Builder Table for PE.

The format of the XLT Key Builder Table definition is common for all blocks as shown in Table 7-201.

Table 7-201. Package Buffer Format for the XLT Key Builder Table

Field	Size (Bytes)	Description
XLT1 Partition Mode	1	This field controls the XLT1 Table MUX selection. Valid values are 0 to 3. All other values are reserved. See Table 7-202. Relevant CSR from "Level-2 profile selection CSRs": GL_<block>_L2PRTMOD.*XLT1*
XLT2 Partition Mode	1	This field controls the XLT2 Table MUX selection. Valid values are 0 to 3. All other values are reserved. See Table 7-203. Relevant CSR from "Level-2 profile selection CSRs": GL_<block>_L2PRTMOD.XLT2
Profile ID Partition Mode	1	This field controls the Profile ID TCAM MUX selection. Valid values are 0 to 3. All other values are reserved. See Table 7-204. Relevant CSR from "Level-2 profile selection CSRs": GL_<block>_PID_L2GKTYPE.*PID_GKTYPE*
Reserved (0)	1	The host must set this value to 0.
Flag15 Mask[31:0]	4	Reduction flag, low 32 bits. Relevant CSR: GL_<block>_FL15_BMPLSB.*BMPLSB*
Flag15 Mask[63:32]	4	Reduction flag, high 32 bits. Relevant CSR: GL_<block>_FL15_BMPMSB.*BMPMSB*
XLT Key Builder Table	192 (8x24)	This table specifies how to construct the Profile ID lookup key for each of 8 possible CDID values, numbered 0 to 7. Each entry is structured as shown in Table 7-205. Relevant CSR from "Non-TCAM Tables": CTLTBL_L2ADDR / CTLTBL_L2DATA

Table 7-202 defines the XLT1 Partition Mode values.

Table 7-202. XLT1 Partition Mode Selection Values

Value	Description
0	XLT1 Table lookup value is metadata[12:0].
1	XLT1 Table lookup value is composed as XLT1_AdSel[0] as the most significant bit, plus the selected metadata[11:0] as the least significant bits.
2	XLT1 Table lookup value is composed as XLT1_AdSel[1:0] as the most significant bits, plus the selected metadata[10:0] as the least significant bits.
3	XLT1 Table lookup value is composed as XLT1_AdSel[2:0] as the most significant bits, plus the selected metadata[9:0] as the least significant bits.
Other	Reserved.

Table 7-203 defines the XLT2 Partition Mode values.

Table 7-203. XLT2 Partition Mode Selection Values

Value	Description
0	XLT2 Table lookup value is metadata[9:0].
1	XLT2 Table lookup value is composed as XLT2_AdSel[0] as the most significant bit, plus the selected metadata[8:0] as the least significant bits.
2	XLT2 Table lookup value is composed as XLT2_AdSel[1:0] as the most significant bits, plus the selected metadata[7:0] as the least significant bits.
3	XLT2 Table lookup value is composed as XLT2_AdSel[2:0] as the most significant bits, plus the selected metadata[6:0] as the least significant bits.
Other	Reserved.

Table 7-204 defines the Profile ID TCAM Partition Mode values. For modes 1,2 and 3, the CDID values is encoded as a one-hot bitmap.

Table 7-204. Profile ID TCAM Partition Mode Selection Values

Value	Description
0	Bits 39:32 of the Profile ID TCAM match value are bits 15:8 of XLT2 Table output.
1	Bits 39:38 of the Profile ID TCAM match value are a one-hot encoding of CDID[0]. Bits 37:32 of the Profile ID TCAM match value are bits 13:8 of XLT2 Table output.
2	Bits 39:36 of the Profile ID TCAM match value are a one-hot encoding of CDID[1:0]. Bits 35:32 of the Profile ID TCAM match value are bits 11:8 of XLT2 Table output.
3	Bits 39:32 of the Profile ID TCAM match value are a one-hot encoding of CDID[2:0].
Other	Reserved.

Table 7-205 defines the XLT Key Builder Table entries.

Table 7-205. Format for XLT Key Builder Table Entries

Field	Size (Bits)	Bit Offset	Description
Reserved	32	0	Reserved for programming tools use. Ignored by firmware.
XLT1 AdSel	3	32	If used, value driven on XTL1 AdSel lines. Relevant CSR: GL_<block>_CTLTBL_L2DATA[2:0][1] (*LINE_OFF*=0)
XLT2 AdSel	3	35	If used, value driven on XTL2 AdSel lines. Relevant CSR: GL_<block>_CTLTBL_L2DATA[5:3] (*LINE_OFF*=0)
Flag0 Select	9	38	Profile ID flags select (same for Flags 1-14 in following rows) Relevant CSR: GL_<block>_CTLTBL_L2DATA[14:6] (*LINE_OFF*=0)
Flag1 Select	9	47	Relevant CSR: GL_<block>_CTLTBL_L2DATA[23:15] (*LINE_OFF*=0)
Flag2 Select	9	56	Relevant CSRs: GL_<block>_CTLTBL_L2DATA[31:24] (*LINE_OFF*=0) GL_<block>_CTLTBL_L2DATA[0:0] (*LINE_OFF*=1)
Flag3 Select	9	65	Relevant CSR: GL_<block>_CTLTBL_L2DATA[9:1] (*LINE_OFF*=1)
Flag4 Select	9	74	Relevant CSR: GL_<block>_CTLTBL_L2DATA[18:10] (*LINE_OFF*=1)
Flag5 Select	9	83	Relevant CSR: GL_<block>_CTLTBL_L2DATA[27:19] (*LINE_OFF*=1)
Flag6 Select	9	92	Relevant CSRs: GL_<block>_CTLTBL_L2DATA[31:28] (*LINE_OFF*=1) GL_<block>_CTLTBL_L2DATA[4:0] (*LINE_OFF*=2)
Flag7 Select	9	101	Relevant CSR: GL_<block>_CTLTBL_L2DATA[13:5] (*LINE_OFF*=2)
Flag8 Select	9	110	Relevant CSR: GL_<block>_CTLTBL_L2DATA[22:14] (*LINE_OFF*=2)
Flag9 Select	9	119	Relevant CSR: GL_<block>_CTLTBL_L2DATA[31:23] (*LINE_OFF*=2)
Flag10 Select	9	128	Relevant CSR: GL_<block>_CTLTBL_L2DATA[8:0] (*LINE_OFF*=3)
Flag11 Select	9	137	Relevant CSR: GL_<block>_CTLTBL_L2DATA[17:9] (*LINE_OFF*=3)
Flag12 Select	9	146	Relevant CSR: GL_<block>_CTLTBL_L2DATA[26:18] (*LINE_OFF*=3)
Flag13 Select	9	155	Relevant CSRs: GL_<block>_CTLTBL_L2DATA[31:27] (*LINE_OFF*=3) GL_<block>_CTLTBL_L2DATA[3:0] (*LINE_OFF*=4)
Flag14 Select	9	164	Relevant CSR: GL_<block>_CTLTBL_L2DATA[12:4] (LINE_OFF=4)
Reserved	9	173	Reserved (0).

Table 7-205. Format for XLT Key Builder Table Entries [continued]

Field	Size (Bits)	Bit Offset	Description
XLT2 MdSel	5	182	XLT1 metadata selection Relevant CSR: GL_<block>_CTLTBL_L2DATA[26:22] (*LINE_OFF*=4) ***Note:*** *XLT2/1 MdSel* are in reverse order from *AdSel* fields
XLT1 MdSel	5	187	XLT1 metadata selection Relevant CSR: GL_<block>_CTLTBL_L2DATA[31:27] (*LINE_OFF*=4) ***Note:*** *XLT2/1 MdSel* are in reverse order from *AdSel* fields.

1. Set GL_<*block*>_CTLTBL_L2ADDR.*AUTO_INC* when writing data to GL_<*block*>_CTLTBL_L2DATA.

Note: Initial value for GL_<block>_CTLTBL_L2ADDR should be 0x1 << 31 (*AUTO_INC* field set). Later, only *LINE_OFF* field of this register should be changed.

7.11.12.2.5 Package Format of XLT1 Tables

This section specifies the package format for configuring the per-block XLT1 table. The XLT1 Table provides an intermediate stage during calculation of a Profile ID. The use of the XLT1 table is block specific, but can, for example, be used to generate a Packet Group ID.

Each pipeline block uses a distinct XLT1 Table configuration and distinct Package Section Type Numbers to contain the configuration data. Table 7-206 shows the section type enumeration. Because the size of the XLT1 Table exceeds the 4 KB maximum size for an AQ command buffer, the host must issue more than one AQ command to complete initialization of the entire table.

Table 7-206. Package Section Type Numbers for XLT1 Tables

Section Type Number	Allowed in Update Package?	Description
12	Yes	XLT1 Table for the Switch.
22	Yes	XLT1 Table for the ACL.
32	Yes	XLT1 Table for the Flow Director.
42	Yes	XLT1 Table for RSS.
82	Yes	XLT1 Table for PE.

The package format for XLT1 Table configuration is common for all blocks as shown in Table 7-207.

Table 7-207. Package Format for XLT1 Table Configuration

Field	Size (Bytes)	Description
Count	2	Number of entries in this section. If the XLT1 table is split into multiple sections, this field describes only the number of entries contained in this section.
Offset	2	Byte offset from the start of the table to the first value in this section. The maximum effective offset for any value is 8191. Effective offsets greater than 8191 are reserved. Relevant CSR: GL_<block>_XLT1_L2ADDR.*LINE_IDX* Should be set together with *AUTO_INC* bit.

Table 7-207. Package Format for XLT1 Table Configuration [continued]

Field	Size (Bytes)	Description
Value	1	Value for the offset specified by the *Offset* field. Relevant CSR from "Non-TCAM Tables": GL_<block>_XLT1_L2DATA.*DATA*
...	...	Subsequent consecutive values.

7.11.12.2.6 Package Format of XLT2 Tables

This section specifies the package format for configuring the per-block XLT2 table. The XLT2 Table provides an intermediate stage during calculation of a Profile ID. The use of the XLT2 table is block specific, but can, for example, be used to expand a set of metadata fields into additional information.

Each pipeline block uses a distinct XLT2 Table configuration and distinct Package Section Type Numbers to contain the configuration data. Table 7-208 shows the section type enumeration. Because the size of the XLT2 Table does not exceed the 4 KB maximum size for an AQ command buffer, the host can initialize the table in parts or as a single AQ command.

Table 7-208. Package Section Type Numbers for XLT2 Tables

Section Type Number	Allowed in Update Package?	Description
13	Yes	XLT2 Table for the Switch.
23	Yes	XLT2 Table for the ACL.
33	Yes	XLT2 Table for the Flow Director.
43	Yes	XLT2 Table for RSS.
83	Yes	XLT2 Table for PE.

The package format for XLT2 Table configuration is common for all blocks as shown in the Table 7-209.

Table 7-209. Package Format for XLT2 Table Configuration

Field	Size (Bytes)	Description
Count	2	Number of entries in this download.
Offset	2	Byte offset from the start of the table to the first value in this section. The maximum effective offset for any value is 1023. Effective offsets greater than 1023 are reserved. Relevant CSR: GL_<block>_XLT2_L2ADDR.*LINE_IDX*. Should be set together with *AUTO_INC* bit.
Value	2	Value for the offset specified by the *Offset* field. Relevant CSR from "Non-TCAM Tables": GL_<block>_XLT2_L2DATA.*DATA*
...	...	Subsequent consecutive values.

7.11.12.2.7 Package Format of Profile ID TCAMs

This section specifies the package format for configuring the per-block Profile ID TCAM. The Profile ID TCAM provides the Profile ID based on the output of the XLT1 Table, the XLT2 Table, Flags, and the CDID. The use of the Profile ID TCAM is block-specific and directs block-specific functionality.

Each pipeline block uses a distinct Profile ID TCAM configuration and distinct Package Section Type Numbers to contain the configuration data. Table 7-210 shows the section type enumeration. Because the size of the Profile ID TCAM can exceed the 4 KB maximum size for an AQ command buffer, the host might have to issue multiple AQ command to initialize the entire table.

Table 7-210. Package Section Details for Profile ID TCAMs

Section Type Number	Allowed in Update Package?	Number of TCAM Entries	Number of Profile IDs	Description
14	Yes	512	256	Profile ID TCAM for the Switch.
24	Yes	512	128	Profile ID TCAM for the ACL.
34	Yes	512	128	Profile ID TCAM for the Flow Director.
44	Yes	512	128	Profile ID TCAM for RSS.
84	Yes	64	32	Profile ID TCAM for PE.

The package format for Profile ID TCAM configuration is common for all blocks as shown in Table 7-211.

Table 7-211. Package Format for Profile ID TCAM Configuration

Field		Size (Bytes)	Description
Count		2	Number of entries in this download.
Entry 0	Address	2	Address of this entry in the TCAM. Relevant CSR from "TCAM Programming": GL_<block>EXT_TCAM_L2ADDR
	Key/Key Invert	10	[79:40] = Encoded Key Invert for this entry. The key is composed of 40 input bits encoded according to Table 7-81. The layout is shown in Table 7-212. Relevant CSR from "TCAM Programming": GL_<block>EXT_TCAM_L2DATALSB.*DATALSB* (bytes 0-3) GL_<block>EXT_TCAM_L2DATAMSB.*DATAMSB* (byte 4) [39:0] = Encoded Key for this entry. The key is composed of 40 input bits encoded according to Table 7-81. The layout is shown in Table 7-212. Relevant CSR from "TCAM Programming": GL_<block>EXT_TCAM_L2DATALSB.*DATALSB* (bytes 0-3) GL_<block>EXT_TCAM_L2DATAMSB.*DATAMSB* (byte 4)
	Profile ID	1	Profile ID output value corresponding to this key. Firmware reserves the Profile ID value of 1 as the default "no match" value. GL_<BLOCK>EXT_K2N_L2ADDR.*LINE_IDX* should be set to (Address /4). GL_<BLOCK>EXT_K2N_L2DATA.*DATA*[Address % 4] should be set to Profile ID.
...		Nx13	All other entries.

Table 7-212 shows the format of the 10-byte Profile Key. The Profile Key occupies Bits 79-0 of the Profile ID TCAM Entry shown in Table 7-212.

Table 7-212. Format of 80-Bit Profile ID TCAM Key Field

Bits 39-32	Bits 31-24	Buts 23-16	Bits 15-0
Eight bits as specified in Table 7-204.	Eight bits produced by the XLT2 table	Eight bits produced by the XLT1 table	16 Flags

7.11.12.2.8 Package Format of Profile ID Redirection Tables

After the Profile ID TCAM generates the Profile ID, the Profile ID Redirection Table can change the Profile ID to a different value. The Profile ID Redirection Table controls this remapping. The Profile ID Redirection Table always maps the input Profile ID to an output Profile ID. Therefore, to allow Profile ID values to pass through unchanged, the host must initialize this table to the trivial identity mapped case (for example, Profile ID 7 maps to Profile ID 7).

Each block in the pipeline contains its own Profile ID Redirection Table. Table 7-213 shows the relevant Package Section Type for each block.

Table 7-213. Package Section Type Numbers for Profile ID Redirection Table

Section Type Number	Allowed in Update Package?	Number of Entries	Description
15	Yes	256	Profile ID Redirection Table for the Switch.
25	Yes	128	Profile ID Redirection Table for the ACL.
35	Yes	128	Profile ID Redirection Table for the Flow Director.
45	Yes	128	Profile ID Redirection Table for RSS.
85	Yes	32	Profile ID Redirection Table for PE.

The package format for Profile ID Redirection Table is common for all blocks as shown in Table 7-214. The size of the table varies depending on the number of Profile ID values supported by particular block.

Table 7-214. Package Format for Profile ID Redirection Table

Field	Size (Bytes)	Description
Count	2	Number of entries in this section.
Offset	2	Offset to the first entry (N) in this section. Offset must be in range 0 to the maximum number of profiles supported by the block as shown in Table 7-213.
Profile ID Redirect Value	1	Redirected value for the Profile ID specified by the *Offset* field. Relevant CSR: GL_<block>_N2N_L2DATA.*DATA0*
...	Nx1	All subsequent entries in consecutive order.[1]

1. Note for firmware: Initial value for GL_<*block*>_N2N_L2ADDR should be 0x1 << 31 (*AUTO_INC* field set).

7.11.12.3 Dynamic Configuration of Profile IDs

A host can program new Profile ID values at runtime. For this purpose, the host issues one or more Update Package AQ commands. The Update Package commands download an incremental package update to the Profile ID logic using the same package buffer formats described in the previous sections.

The host can change all aspects of Profile ID configuration dynamically via the Update Package AQ command. Firmware can provide some error checking, but responsibility for correct operation lies with the host software.

7.11.12.3.1 Field Vector Extraction Table

After the Profile ID Redirection Table generates the Profile ID, the field vector extraction logic takes the Profile ID as input and produces a field vector. The Field Vector Table configures this extraction logic.

Table 7-215. Package Section Type Numbers for Field Vector Extraction Tables

Section Type Number	Allowed in Update Package?	Number of Entries	FV Size (32-bit Words)	Description
16	Yes	256	48	Field Vector Table for the Switch.
26	Yes	128	32	Field Vector Table for the ACL.
36	Yes	128	24	Field Vector Table for the Flow Director.
46	Yes	128	24	Field Vector Table for RSS.
86	Yes	32	32	Field Vector Table for the PE.
99	No	1	32	Field Vector Table for Manageability. ***Note:*** PFR has no effect on this resource.

The number of 16-bit words available in a Field Vector varies by hardware block as specified by the Section Type Number. Regardless, all blocks use a common format as shown in Table 7-216.

Table 7-216. Package Format for Profile ID Redirection Table

Field	Extraction Word Fields	Size (Bits)	Description
Count		16	Number of entries in this section.
Offset (Base Profile ID)		16	Offset to the first table entry in this section. The offset value is the same as the logical Profile ID. Subsequent entries in this section occupy consecutive Profile ID values in the FV extraction table. Relevant CSR: GL_<block>_PRFLM_CTRL[n].*PRFL_IDX*

Table 7-216. Package Format for Profile ID Redirection Table [continued]

Field	Extraction Word Fields	Size (Bits)	Description
Entry 0 (for first Profile ID)	Word 0 Protocol ID	8	Protocol ID for extraction word 0. Relevant CSR: GL_<block>_PRFLM_DATA_[n][Word Offset].*PROT*
	Word 0 Offset	9	Byte offset for extraction word 0. Relevant CSR: GL_<block>_PRFLM_DATA_[n][Word Offset].*OFF*
	Reserved	15	Reserved (0).
	Word 1 Protocol ID	8	Protocol ID for extraction word 1.
	Word 1 Offset	9	Byte offset for extraction word 1.
	Reserved	15	Reserved (0).
	...		All intervening words.
	Word N Protocol ID	8	Last Protocol ID and Byte Offset for this FV entry, where N is one less than the number of 16 bit words in the FV. See Table 7-215 for the size of the FV per block.
	Word N Offset	9	
	Reserved	15	
All other entries		Size Varies	Each subsequent entry occupies placed at previous Profile ID plus one.

7.11.12.4 VSI Lists

The firmware reserves VSI Prune lists 0 through 7 as empty VSI lists.

7.11.12.5 Recipe Configuration

This section specifies the recipes for packet processing in the Switch block. At a minimum, a complete package must contain a recipe with a default action 'forward to VSI list 0'.

Table 7-217. Package Section Type Numbers for Recipe Configuration

Section Type Number	Allowed in Update Package?	Description
92	No	Recipe configuration for the Switch.

Table 7-218. Package Format of the Recipe Configuration Section

Field		Size (Bits)	Bit Offset	Description
Count		16	0	Number of entries in this section. Valid values 1-63.
Reserved		16	16	Reserved (0).
Entry 0	Recipe ID	8	32	The unique, incrementing Recipe ID automatically assigned by the tools.
	Reserved	24	40	Reserved (0).
	Dependent Recipes	64	64	A bitmap representation of the dependent Recipe IDs.
	Recipe	224	128	The 28-byte recipe content, as described in Table 7-9.
...		Nx320	352	All consecutive entries, 40 bytes each.

7.11.12.6 Recipe-to-Profile Associations

This section specifies the associations between profiles and recipes for packet processing in the Switch block.

Table 7-219. Package Section Type Numbers for Recipe-to-Profile Associations

Section Type Number	Allowed in Update Package?	Description
93	No	Recipe to Profile associations for the Switch.

Table 7-220. Package Format of the Recipe Configuration Section

Field		Size (Bits)	Bit Offset	Description
Count		16	0	Number of entries in this section. Valid values 1-63.
Offset (Base Profile ID)		16	16	Offset to the first table entry in this section. The offset value is the same as the logical Profile ID. Subsequent entries in this section occupy consecutive Profile ID values.
Entry 0	Recipe Associations	64	32	Bitmap of the recipe indexes associated with this profile. The bitmap should include both root and non-root recipes.
...		Nx8	96	All consecutive entries, eight bytes each.

7.11.12.6.1 Miscellaneous Profile and Control Domain Configuration

This section specifies the miscellaneous configuration section for profile and control domains.

Table 7-221. Package Section Type Numbers for Miscellaneous Profile and Control Domain Configuration

Section Type Number	Allowed in Update Package?	Description
100	Yes	Miscellaneous Profile and Control Domain Configuration for Switch.
101	Yes	Miscellaneous Profile and Control Domain Configuration for ACL.
102	Yes	Miscellaneous Profile and Control Domain Configuration for Flow Director.
103	Yes	Miscellaneous Profile and Control Domain Configuration for RSS.
104	Yes	Miscellaneous Profile and Control Domain Configuration for PE.

Table 7-222. Package Format of the Miscellaneous Profile and Control Domain

Field	Size (Bits)	Description
Static CDID Selection	32	Enable static CDID selection, and set the static CDID. The value of all FFs is reserved for internal use by tools before conversion to a valid value. [31] = Enable static CDID selection. [30:4] = Reserved (0). [3:0] = Static CDID value. Relevant CSR: GL_*EXT_FORCE_L1CDID

Table 7-222. Package Format of the Miscellaneous Profile and Control Domain [continued]

Field	Size (Bits)	Description
MDID Digest Mask	64	The MDID digest mask. The value of all 0 is reserved for internal use by tools before conversion to a valid value. [63:48] = Reserved (0). [47:32] = The 16 MSBs for the MDID digest mask. [31:0] = The 32 LSBs for the MDID digest mask. Relevant CSRs: GL_*EXT_L2_PMASK0, GL_*EXT_L2_PMASK1
TCAM Input Mask	64	The TCAM input mask. The value of all 0 is reserved for internal use by tools before conversion to a valid value. [63:40] = Reserved (0). [39:32] = The eight MSBs for the TCAM input mask. [31:0] = The 32 LSBs for the TCAM input mask. Relevant CSRs: GL_*EXT_L2_TMASK0, GL_*EXT_L2_TMASK1
Reserved	96	Reserved (0)

7.11.12.6.2 Mask Select Filters

This section specifies the mask select filters section.

Table 7-223. Package Section Type Numbers for Mask Select Filters Configuration

Section Type Number	Allowed in Update Package?	Description
105	No	Mask Select Filters for RSS.
106	No	Mask Select Filters for FD.
107	No	Mask Select Filters for PE.

Table 7-224. Package Format of Mask Select Filters

Field		Size (Bits)	Description
Count		16	Number of entries in this section.
Offset		16	Offset to the first Profile ID in this section. Valid values are 0..31 in the PE block and 0..127 in FD and RSS blocks.
Entry 0	Mask Select	32	The bitmap of registers to be enabled for the given Profile ID. Relevant CSR: GLQF_[FD|H|PE]MASK_SEL
---		Nx32	All consecutive entries, four bytes each.

7.11.12.6.3 Quad Hash Control Table

This section specifies the quad hash control table section.

Table 7-225. Package Section Type Numbers for Quad Hash Control Table

Section Type Number	Allowed in Update Package?	Description
108	No	Quad Hash Control Table for PE.

Table 7-226. Package Format of Mask Select Filters

Field		Size (Bits)	Description
Count		16	Number of entries in this section.
Offset		16	Offset to the first Profile ID in this section. Valid values are 0..31.
Entry 0	TO_QH	2	The target of given profile: 00b = Reserved. 01b = Packet is candidate for QH filter. 10b = Packet is candidate for the PE bypassing the QH filter. 11b = Packet is candidate for the QH filter and forward to the PE even if it does not match the filter. Relevant CSR: GLQF_PE_CTL2
	APBVT	1	Packet profile 'n' should hit the APBVT in order to be a candidate for the QH or directly to the PE. Relevant CSR: GLQF_PE_CTL2
	Reserved	29	Reserved (0)
...		Nx32	All consecutive entries, four bytes each.

7.11.13 Parser Configuration

7.11.13.1 Introduction

Parser logic performs initial packet analysis and generates metadata for use in subsequent pipeline stages. The packet processing pipeline contains two Parsers, one for the receive path (Rx Parser) and one for the transmit path (Tx Parser). Both Rx and Tx Parsers share the same design.

A parser is not a monolithic structure, but consists of many interdependent sub-components. A package contains dedicated sections for each of these sub-component within the Parser.

7.11.13.2 Parse Graph

The Parse Graph logic is the principle driver of the packet analysis process within the Parser. The Parse Graph system consists of a variety of tables, each with its package section. See Table 7-227.

Table 7-227. Package Section Type Numbers for Parse Graph CAMs

Section Type Number	Allowed in Update Package?	Description
50	No	Parse Graph CAM for the Rx Parser.
60	No	Parse Graph CAM for the Tx Parser.

7.11.13.2.1 CAM Entry Algorithm

During a CAM query, hardware uses a hash algorithm to select an entry index, then checks subsequent entries for a match. The algorithm is as follows:

1. Calculate the entry index from 0-max index using the hash algorithm on the lookup key. The maximum index for the Parse Graph CAM is 2047. The maximum index for the No-Match CAM is 1024.
2. Starting from the calculated entry index, check if the entry at index matches the key. If the entry matches return the corresponding value. If the entry does not match, iterate to the next entry up to the maximum count for the CAM. The Parse Graph CAM checks up to eight consecutive entries for a match. The No-Match CAM checks up to four consecutive entries for a match.
3. If no entry matched the key, query the Spill CAM starting at entry 0. If the Spill CAM contains a match, return the corresponding value. If the Spill CAM does not have a match, no match was found.

7.11.13.2.2 Determining CAM Entry Index

This section provides a pseudo-code algorithm to determine the correct CAM entry index for a given PG key. This algorithm produces the same result as the hardware algorithm.

The Parser Miscellaneous Configuration section (see Section 7.11.13.16) specifies the seed value used to compute the CAM Entry Index.

```
// bitset<N> is a collection of N bits
// myset[i] references the i'th bit of the biset 'myset'
// & is a bitwise logical AND operation
// ^ is a bitwise logical XOR operation
// << is a bitwise shift left with zero fill operation
```

```
bitset<11> GetPGCAMEntryIndex(bitset<72> Key, bitset<83> Seed)
    bitset<11> Hash;         // Computed hash value
    bitset<83> PadKey = Key; // Key in bottom 72 bits
    PadKey = PadKey << 11;   // Shift Key to top 72 bits
    for i = 0 to 11 do
        bitset<84> Temp = PadKey & (Seed << i)
        bitset<1> TempBit = Temp[0]
        for j = 1 to 84 do
            TempBit = TempBit ^ Temp[j]
        Hash[10-i] = TempBit
    return Hash
bitset<10> GetNoMatchCAMEntryIndex(bitset<40> Key, bitset<51> Seed)
    bitset<10> Hash; // Computed hash value
    bitset<51> PadKey = Key; // Key in bottom 40 bits
    PadKey = PadKey << 11; // Shift Key to top 40 bits
    for i = 0 to 10 do
        bitset<50> Temp = PadKey & (Seed << i)
        bitset<1> TempBit = Temp[0]
        for j = 1 to 50 d
            TempBit = TempBit ^ Temp[j]
        Hash[10-i] = TempBit
    return Hash
```

7.11.13.2.2.1 CAM Entry Index Examples

PG No-Match CAM slice entry index hash examples:

```
Input Seed = 0x1b0ce7ae9030f
Input Key = 0x0000008000
Output Hash = 0x2e9

Input Seed = 0x1b0ce7ae9030f
Input Key = 0x0000298808
Output Hash = 0x378
```

PG CAM slice entry index hash examples:

```
Input Seed = 0x05c5666c1b0ce7ae9030f
Input Key = 0x000000040000008000
Output Hash = 0x2df

Input Seed = 0x05c5666c1b0ce7ae9030f
Input Key = 0x0000002c00001a800d
Output Hash = 0x6d7
```

7.11.13.2.3 Determining CAM Slice

The entry is placed in the unoccupied entry at the calculated entry index. If an entry is already occupied, subsequent entries are searched in order up to the maximum for the CAM (0 to 7 for the PG CAM, and 0 to 3 for the No-Match CAM).

If the entry and all subsequent entries are occupied up to the maximum count for the CAM, the entry is placed in the corresponding Spill CAM at the first available location. If the Spill CAM is already full, the result is an error condition in which the entry cannot be placed.

7.11.13.2.4 Package Format of the Parse Graph CAM

The hub of the Parse Graph logic is the Parse Graph CAM. Table 7-228 shows the package buffer layout of the Parse Graph CAM.

Table 7-228. Package Format for Parse Graph CAM Configuration

Field	Size (Bytes)	Description
Count	2	Number of entries in this section.
Offset	2	Offset to the first entry in this section.All entries are contiguous from this offset. All entries must be in range 0 to 2047. Entries with an offset greater than 2047 are reserved. Relevant CSRs: GLGEN_ANA_CFG_CTRL.*LINE_IDX* = Offset >> 3 GLGEN_ANA_CFG_CTRL.*TABLE_ID* = 4 + (Offset % 8)
Entry 0	16	First CAM entry. Each CAM entry contains a key and value. Each entry is formatted as shown in Table 7-229. Relevant CSR uses Parser indirect interface with Table IDs 4 to 11 (PG mem0 to PG mem7). GLGEN_ANA_CFG_WRDATA.*WR_DATA*, 32 bits/write
---	Nx16 + 4	All other entries.

Table 7-229 shows the format of each 16-byte Parse Graph CAM entry.

Table 7-229. Package Format of the Parse Graph CAM Entry

	Field	Width (Bits)	Bit Offset	Description
Key (73 bits)	Valid	1	0	The parse graph entry is valid. The tools set this bit to 0 for undefined nodes in the parse graph.
	Node ID	11	11:1	Node ID of the protocol in the current analysis round.
	Flag 0	1	12	Optional Flag 0.
	Flag 1	1	13	Optional Flag 1.
	Flag 2	1	14	Optional Flag 2.
	Flag 3	1	15	Optional Flag 3.
	Boost Hit	1	16	Flag set when the Boost TCAM Search resulted in a match. Clear otherwise.
	Boost Index	8	24:17	Boost TCAM match index if Boost Hit Flag == 1. This field is zero otherwise.
	ALU Reg	16	40:25	Optional ALU register value if enabled. Zero otherwise.
	Next Proto Key	32	72:41	Next Protocol value extracted from the packet.

Table 7-229. Package Format of the Parse Graph CAM Entry [continued]

	Field	Width (Bits)	Bit Offset	Description
Action (55 bits)	Next Node ID	11	83:73	Node ID used for the next analysis round.
	Next PC	8	91:84	IMEM Program Counter used for the next analysis round.
	Is Protocol Group	1	92	Set if the Protocol ID value is a protocol group ID. Clear if the Protocol ID value is a Protocol ID. See Section 7.11.13.9.
	Reserved	3	95:93	Reserved (0).
	Protocol ID	8	103:96	Protocol ID (Is Protocol Group == 0) or Protocol Group ID (Is Protocol Group = 1)
	Is Marker Group	1	104	Set if the Marker ID value is a Marker Group ID. Clear if the Marker ID value is a Marker ID. See Section 7.11.13.10.
	Marker ID	8	112:105	Marker ID (Is Marker Group == 0) or Marker Group ID (Is Marker Group == 1)
	Is Last Round	1	113	Set if this is a last analysis round for this packet. Clear otherwise.
	Header Offset Polarity	1	114	Set if the Header Offset Inc value is added to the current Header Offset value. Clear if the Header Offset Inc value is subtracted from the current Header Offset.
	Header Offset Inc	9	123:115	Number of bytes to add or subtract from the Header Offset.
	Reserved	4	127:124	Reserved (0).

7.11.13.2.5 Package Format of the Parse Graph No-Match CAM

The Parse Graph No-Match logic defines the output values used when the Parse Graph CAM did not find a match for the input key.

Table 7-230. Package Section Type Numbers for Parse Graph No-Match CAMs

Section Type Number	Allowed in Update Package?	Description
51	No	Parse Graph No-Match CAM for the Rx Parser.
61	No	Parse Graph No-Match CAM for the Tx Parser.

Table 7-231 shows the package buffer layout of the PG No Match CAM.

Table 7-231. Package Format for Parse Graph No-Match CAM Configuration

Field	Size (Bytes)	Description
Count	2	Number of entries in this section.
Offset	2	Offset to the first entry in this section. All entries are contiguous from this offset. All entries must be in range 0 to 1023. Entries with an offset greater than 1023 are reserved.

Table 7-231. Package Format for Parse Graph No-Match CAM Configuration [continued]

Field	Size (Bytes)	Description
Entry 0	12	First CAM entry. Each CAM entry contains a key and value. Each entry is formatted as shown in Table 7-232. Relevant CSR uses Parser indirect interface with Table IDs 21 to 24 (no match PG mem0 to no match PG mem3).
---	Nx12 + 4	All other entries.

Table 7-232 shows the format of each 12-byte PG No-Match CAM entry. The format is identical to the Parse Graph CAM format, with the exception that the *Next Proto Key* field is not specified.

Table 7-232. Package Format of the Parse Graph CAM Entry

	Field	Width (Bits)	Bit Offset	Description
Key (41 bits)	Valid	1	0	The parse graph entry is valid. The tools set this bit to 0 for undefined nodes in the parse graph.
	Node ID	11	11:1	Node ID of the protocol in the current analysis round.
	Flag 0	1	12	Optional Flag 0.
	Flag 1	1	13	Optional Flag 1.
	Flag 2	1	14	Optional Flag 2.
	Flag 3	1	15	Optional Flag 3.
	Boost Hit	1	16	Flag set when the Boost TCAM Search resulted in a match. Clear otherwise.
	Boost Index	8	24:17	Boost TCAM match index if Boost Hit Flag == 1. This field is zero otherwise.
	ALU Reg	16	40:25	Optional ALU register value if enabled. Zero otherwise.
Action (55 bits)	Next Node ID	11	51:41	Node ID used for the next analysis round.
	Next PC	8	59:52	IMEM Program Counter used for the next analysis round.
	Is Protocol Group	1	60	Set if the Protocol ID value is a protocol group ID. Clear if the Protocol ID value is a Protocol ID.
	Reserved	3	63:61	Reserved(0)
	Protocol ID	8	71:64	Protocol ID (Is Protocol Group == 0) or Protocol Group ID (Is Protocol Group = 1)
	Is Marker Group	1	72	Set if the Marker ID value is a Marker Group ID. Clear if the Marker ID value is a Marker ID.
	Marker ID	8	80:73	Marker ID (Is Marker Group == 0) or Marker Group ID (Is Marker Group == 1)
	Is Last Round	1	81	Set if this is a last analysis round for this packet. Clear otherwise.
	Header Offset Polarity	1	82	Set if the Header Offset Inc value is added to the current Header Offset value. Clear if the Header Offset Inc value is subtracted from the current Header Offset.
	Header Offset Inc	9	91:83	Number of bytes to add or subtract from the Header Offset.
	Reserved	4	95:92	Reserved (0)

7.11.13.3 PG Spill CAM

The PG Spill CAM holds PG CAM entries that could not be placed in any CAM slice due to a hash collision.

Table 7-233. Package Section Type Numbers for PG Spill CAMs

Section Type Number	Allowed in Update Package?	Description
76	No	PG Spill CAM for the Rx Parser.
77	No	PG Spill CAM for the Tx Parser.

Table 7-234 shows the format of the PG Spill CAM section.

Table 7-234. PG Spill CAM Section Format

Field		Size (Bits)	Description
Count		16	Number of Key-Value pairs in this section.
Offset		16	Offset to the first entry in this section. All entries are contiguous from this offset. All entries must be in range 0 to 127. Entries with an offset greater than 127 are reserved. Relevant CSR uses Parser indirect interface, *Table Address* field.
Entry 0	Action	56	Parse Graph output value contains 55 bits as shown in Table 7-229 and one reserved bit (0). Relevant CSR uses Parser indirect interface with Table ID 2 (PG spill buffer action).
	Key	80	Match key for this PG entry contains 74 inputs bits as shown in Table 7-229 and seven reserved bits (0). Relevant CSR uses Parser indirect interface with Table ID 3 (PG spill buffer key).
---		Nx136 + 32	All other entries.

7.11.13.4 Parse Graph No-Match Spill CAM

The No-Match Spill CAM holds No-Match CAM entries that could not be placed in any CAM slice due to a hash collision.

Table 7-235. Package Section Type Numbers for Parse Graph No-Match Spill CAMs

Section Type Number	Allowed in Update Package?	Description
78	No	No-Match Spill CAM for the Rx Parser.
79	No	No-Match Spill CAM for the Tx Parser.

Table 7-236 shows the format of the No-Match Spill CAM section.

Table 7-236. No-Match Spill CAM Section Format

Field		Size (Bits)	Description
Count		16	Number of Key-Value pairs in this section.
Offset		16	Offset to the first entry in this section. All entries are contiguous from this offset. All entries must be in range 0 to 63. Entries with an offset greater than 63 are reserved. Relevant CSR uses Parser indirect interface, *Table Address* field.
Entry 0	Action	56	No-Match output value contains 55 bits as shown in Table 7-232 and one reserved bit (0). Relevant CSR uses Parser indirect interface with Table ID 19 (No-Match PG spill buffer action).
	Key	48	Match key for this PG entry contains 41 inputs bits as shown in Table 7-232 and seven bits reserved (0). Relevant CSR uses Parser indirect interface with Table ID 20 (No-Match PG spill buffer key).
---		Nx104 + 32	All other entries.

7.11.13.5 Node PTYPE Table

The Node PTYPE table defines the PTYPE value based on the NEXT_NODE_ID output from the Parse Graph or PG No-Match CAM. The PType value of 0 is defined to be INVALID_PTYPE. Software must set entries in this table to INVALID_PTYPE when the Marker PTYPE TCAM defines the PTYPE of the packet.

Table 7-237. Package Section Type Numbers for Node PTYPE Tables

Section Type Number	Allowed in Update Package?	Description
54	No	Node PTYPE Table for the Rx Parser.
64	No	Node PTYPE Table for the Tx Parser.

Table 7-238 shows the section format for the PTYPE table.

Table 7-238. Node PTYPE Table Section Format

Field	Size (Bytes)	Description
Count	2	Number of entries in this section.
Offset	2	PTYPE table address of the first entry in this section.
Entry 0	2	First PTYPE table entry containing the PTYPE value (0-1023) in bits 9:0. Bit 10 is the error output. All other bits are reserved. A PTYPE value of 0 represents an invalid PTYPE, which causes hardware to use the MARKER_PTYPE TCAM to determine the packet PTYPE. Relevant CSR uses Parser indirect interface with Table ID 15 (node cntx ID).
...	Nx2	All other entries.

7.11.13.6 Marker PTYPE TCAM

The Marker PTYPE TCAM defines the PTYPE value based on the accumulated marker bit vector.

Table 7-239. Package Section Type Numbers for Marker PTYPE TCAMs

Section Type Number	Allowed in Update Package?	Description
55	No	Marker PType Table for the Rx Parser.
65	No	Marker PType Table for the Tx Parser.

Table 7-240 shows the section format for the PTYPE table.

Table 7-240. Marker PTYPE TCAM Section Format

Field		Size (Bytes)	Description
Count		2	Number of Key-Value pairs in this section.
Reserved		2	Reserved (0).
Entry 0	Address	2	Address of this TCAM entry. Valid values are 0-1023. Values greater than 1023 are reserved. Relevant CSR uses Parser indirect interface, *Table Address* field.
	Value	2	PTYPE number (0-1023) associated with this key. Values greater than 1023 are reserved. Relevant CSR uses Parser indirect interface with Table ID 18 (PTYPE TCAM action RAM).
	Key	10	Match key for this PType value. The key contains 80 bits, encoded according to Table 7-81. The lower 72 bits represent 72 possible marker bits. The eight most significant input bits are a one-hot representation of the Control Domain ID. Relevant CSR uses Parser indirect interface with Table ID 17 (PTYPE TCAM key). Each key takes up two lines in the table.
	Key Invert	10	The key invert for this PTYPE value. The key-invert contains 80 input bits, encoded according to Table 7-81. The lower 72 bits represent the key-invert value for 72 possible marker bits. The eight most significant input bits are the key-invert of a one-hot representation of the Control Domain ID. Relevant CSR uses Parser indirect interface with Table ID 17 (PTYPE TCAM key). Each key takes up two lines in the table.
...		Nx24	All other entries.

7.11.13.7 IMEM

7.11.13.7.1 IMEM Section

The Instruction Memory (IMEM) section contains three VLIW instructions for the ALUs, a key extraction instruction for the Parse Graph CAM, and several other fields.

Table 7-241. Package Section Type Numbers for IMEM

Section Type Number	Allowed in Update Package?	Description
52	No	IMEM for the Rx Parser.
62	No	IMEM for the Tx Parser.

Table 7-242 shows the format of an IMEM section.

Table 7-242. IMEM Section Format

Field		Size (Bits)	Bit Offset	Description
Count		16	0	Number of IMEM instructions in this section. Must be at least 1. Values greater than 191 are reserved.
Offset		16	16	IMEM address of the first entry in this section. The effective offset of all IMEM entries in this section must be in range 0-191. Values greater than 191 are reserved. Relevant CSR uses Parser indirect interface, *Table Address* field. Instruction content is written using Parser indirect register interface with Table ID value 14.
Entry 0	Boost Master	4	32	Boost Master flags (see Section 7.11.13.7.7).
	Boost Key Build	10	36	Boost TCAM Key Build (see Section 7.11.13.7.6).
	PG Priority	2	46	Parse Graph resolver priority control (see Section 7.11.13.7.5).
	Next Proto Key Build	18	48	Next Protocol extraction directive (see Section 7.11.13.7.4).
	PG Key Build	35	66	Format directive for the Parse Graph key (see Section 7.11.13.7.3).
	ALU 0 Instruction	96	101	Instruction for ALU 0 (see Section 7.11.13.7.2).
	ALU 1 Instruction	96	197	Instruction for ALU 1 (see Section 7.11.13.7.2).
	ALU 2 Instruction	96	293	Instruction for ALU 2 (see Section 7.11.13.7.2).
	Reserved	27	389	Reserved (0).
...		Nx384		All subsequent entries, 48 bytes each.

7.11.13.7.2 ALU Instructions

The format of an individual ALU instruction is shown in Table 7-243.

Table 7-243. ALU Instruction Format, 3x per IMEM Entry

Field	Width (Bits)	Bit Offset	Description
Opcode	6	5:0	Instruction Opcode. See Table 7-244.
Source Start	8	13:6	Source register start bit, 0-255.
Source Length	5	18:14	Source register length, 0-31.
Shift/Xlate Select	1	19	0b = Shift Source Register by four bits. 1b = Fetch from the Xlate Table using 2-bit translate key.
Shift/Xlate Key	4	23:20	4-bit shift left operand, or 2-bit translate key.
Source Register ID	7	30:24	Source register ID.
Dest. Register ID	7	37:31	Destination register ID.
Inc 0	1	38	When set increment Counter 0.
Inc 1	1	39	When set increment Counter 1.
Protocol Offset Opcode	2	41:40	00b = Protocol Offset does not change. 01b = Protocol Offset = Header Offset + Protocol Offset. 10b = Protocol Offset = Header Offset - Protocol Offset. 11b = Reserved.
Protocol Offset	8	49:42	Number of bytes to add or remove from the protocol offset to calculate the next protocol offset.
Branch Address	8	57:50	Absolute value of Branch Address in IMEM, 0-191.
Immediate	16	73:58	Immediate value.
Dedicated Flags Enable	1	74	When set activates dedicated flags logic. ALU must not be configured to set flags when this bit is set.
Dest. Start	6	80:75	Destination start bit. When *Dedicated Flags Enable* is set, this field specifies that starting bit within the 64-bit Flags metadata. When *Dedicated Flags Enable* is clear, this field specifies a single bit of the destination register written by the SETEQ/NEQ/LT/GT, ANDEQ/NEQ/LT/GT, OREQ/NEQ/LT/GT instructions. Unused in all other cases.
Dest. Length	6	86:81	Destination length in the flags field. Used only when *Dedicated Flags Enable* is set.
Flags Extract Imm.	1	87	0b = Copies Dest. Length flags from packet starting from the bit specified by *Flags Start*. 1b = Copies Dest. Length flags from Flags Immediate field.
Flags Start/Immediate	8	95:88	Specifies the start bit in the packet header from which to extract flags, or an immediate value to copy to flags as controlled by the *Flags Extract Imm* field. Used only when *Dedicated Flags Enable* is set.

The six opcode fields define the ALU instruction as shown in Table 7-244.

Table 7-244. ALU Opcodes

Mnemonic	Opcode
PARK	0
MOV (with ADD expr.)	1
ADD	2
Reserved	3
MOV (with AND expr.)	4
AND	5
AND Immediate	6
MOV (with OR expr.)	7
OR	8
MOV (with XOR expr.)	9
XOR	10
NOP	11

Mnemonic	Opcode
BR	12
BREQ	13
BRNEQ	14
BRGT	15
BRLT	16
BRGEQ	17
BRLEQ	18
SETEQ	19
ANDEQ	20
OREQ	21
SETNEQ	22
ANDNEQ	23

Mnemonic	Opcode
ORNEQ	24
SETGT	25
ANDGT	26
ORGT	27
SETLT	28
ANDLT	29
ORLT	30
MOV (with SUB expr.)	31
SUB	32
Reserved	33-62
Invalid Opcode	63

7.11.13.7.3 Parse Graph Key Build

Each IMEM entry contains a directive for building a portion of the Parse Graph Key. The format of a Parse Graph Key Build directive is shown in Table 7-245.

Table 7-245. Parse Graph Key Build Directive per IMEM and Boost TCAM Entry

Field	Width (Bits)	Bit Offset	Description
Flag 0 Enable	1	0	When set, the flag at Flag 0 Index is copied to the *Flag 0* field of the Parse Graph key. When cleared, hardware set the *Flag 0* field of the key to 0.
Flag 0 Index	6	6:1	Index of the Flag0 component of the Parse Graph key, 0-63.
Flag 1 Enable	1	7	When set, the flag at Flag 1 Index is copied to the *Flag 1* field of the Parse Graph key. When cleared, hardware set the *Flag 1* field of the key to 0.
Flag 1 Index	6	13:8	Index of the Flag1 component of the Parse Graph key, 0-63.
Flag 2 Enable	1	14	When set, the flag at Flag 2 Index is copied to the *Flag 2* field of the Parse Graph key. When cleared, hardware set the *Flag 2* field of the key to 0.
Flag 2 Index	6	20:15	Index of the Flag2 component of the Parse Graph key, 0-63.
Flag 3 Enable	1	21	When set, the flag at Flag 3 Index is copied to the *Flag 3* field of the Parse Graph key. When cleared, hardware set the *Flag 3* field of the key to 0.
Flag 3 Index	6	27:22	Index of the Flag3 component of the Parse Graph key, 0-63.
ALU Register Index	7	34:28	Index of the ALU Register component of the Parse Graph key.

7.11.13.7.4 Next Protocol Key Build

Each IMEM and Boost TCAM entry contains a directive for extracting the next protocol from the packet header. The format of a Next Protocol Key Build directive is shown in Table 7-246.

Table 7-246. Next Protocol Key Build Directive per IMEM Entry

Field	Width (Bits)	Bit Offset	Description
Opcode	2	0	00b = Extract the *Next Protocol* field from the Header Value starting from the bit position specified in the *Start* field and for the number of bits specified in the *Length* field. 01b = Build *Next Protocol* field with the value of the register in the *Reg 0* field in *Next Protocol field* bits 15:0, and the value of the register in the *Reg 1* field in *Next Protocol* field bits 31:16. 10b = Bypass the Parse Graph, in which case the Parse Graph logic does not set the Next Protocol Key value. 11b = Reserved.
Start/Reg 0	8	2	*Start* field (Opcode = 0) or *Reg 0* field (Opcode = 1).
Length/Reg 1	8	10	*Length* field (Opcode = 0) or *Reg 1* field (Opcode = 1).

7.11.13.7.5 PG Priority Control

Each IMEM and Boost TCAM entry contains two bits to controlling how the hardware resolver determines precedence from among multiple sources of register update. The format of a PG Priority Control is shown in Table 7-247.

Table 7-247. PG Priority Control per IMEM and Boost TCAM Entry

Field	Width (Bits)	Description
Priority	2	00b = The resolver uses the following priority: ALU2 is priority 3 (highest) ALU 1 is priority 2 ALU 0 is priority 1 PG is priority 0 (lowest) 01b = The resolver uses the following priority: ALU2 is priority 3 (highest) ALU 1 is priority 2 PG is priority 1 ALU 0 is priority 0 (lowest) 10b = The resolver uses the following priority: ALU2 is priority 3 (highest) PG is priority 2 ALU 1 is priority 1 ALU 0 is priority 0 (lowest) 11b = The Resolver uses the following priority: PG is priority 3 (highest) ALU2 is priority 2 ALU 1 is priority 1 ALU 0 is priority 0 (lowest)

7.11.13.7.6 Boost Key Build

The Boost Key Build field controls the value used as the 8-bit Boost TCAM search key qualifier (TSR). The TSR field contains either an arbitrary 8-bit value, or the content of the TSR register.

Table 7-248. Boost Key Build per IMEM Entry

Field	Width (Bits)	Bit Offset	Description
Priority	8	7:0	Arbitrary 8-bit value used when the *TSR Control* bit is 0.
TSR Control	2	9:8	Control bit selecting the value used the Boost TCAM search key qualifier. 00b = The Boost TCAM search key qualifier is the value in bits 7:0. 01b = The Boost TCAM search key qualifier is the content of the TSR register. 10b = Reserved. 11b = Reserved.

7.11.13.7.7 Boost Master

This field contains four flags controlling the user of Boost TCAM output. For each bit set, Boost TCAM output overrides IMEM output for the corresponding hardware block. This override occurs only in the case of a Boost lookup match. When a Boost lookup fails, IMEM output is used.

For each 0 bit, the corresponding hardware block ignores Boost TCAM output, even in the case of a Boost lookup match.

Table 7-249. Boost Master Field

Bit	Description
0	ALU 0
1	ALU 1
2	ALU 2
3	Parse Graph

7.11.13.8 Boost TCAM

The Boost TCAM provides a wide 160-bit match on a combination of packet header and TSR bits. The least significant 152 bits match against the packet header, starting from the current header offset. The most significant eight bits match the selected TSR (see Section 7.11.13.7.6).

Table 7-250. Package Section Type Numbers for the Boost TCAMs

Section Type Number	Allowed in Update Package?	Description
56	Yes	Boost TCAM for the Rx Parser.
66	Yes	Boost TCAM for the Tx Parser.

The Boost TCAM contains 256 entries, with a format similar to IMEM content as shown in Table 7-251.

Table 7-251. Format of the Boost TCAM Section

Field		Size (Bits)	Bit Offset	Description
Count		16	0	Number of entries in this section. Must be at least 1.
Reserved		16	16	Reserved (0).
Entry 0	Address	16	32	Address of this entry in the TCAM, 0-255. Smaller numbers imply higher priority. Values > 255 are reserved. Relevant CSR is the *Table Address* field of the Parser indirect interface.
	Reserved	16	48	Reserved (0).
	Key/Key Invert	320	32	[319:160] = The key-invert used to match this TCAM entry. The key invert consists of 160 input bits, encoded according to Table 7-81. Relevant CSR uses the Parser indirect interface with Table ID 0 (TCAM key). Each key takes up two lines in the table. [159:0] = The key used to match this TCAM entry. The key consists of 160 input bits, encoded according to Table 7-81. The TCAM matches the lower 152 bits of the key against the next 152 bits of the packet header (%HDR register) starting from the current header offset (%HO register). The upper eight bits of the input value are matched against the TCAM Search Key (%TSR register). Relevant CSR uses the Parser indirect interface with Table ID 0 (TCAM key). Each key takes up two lines in the table.
	Boost Hit Index Group	8	384	Provides the Boost TCAM Hit Index part of the Parse Graph Key. Relevant CSR uses the Parser indirect interface with Table ID 13 (TCAM action RAM).
	PG Priority	2	392	Parse Graph resolver priority control (see Section 7.11.13.7.5). Relevant CSR uses the Parser indirect interface with Table ID 13 (TCAM action RAM).
	Next Proto Key Build	18	394	Next Protocol extraction directive (see Section 7.11.13.7.4). Relevant CSR uses the Parser indirect interface with Table ID 13 (TCAM action RAM).
	PG Key Build	35	412	Format directive for the Parse Graph key (see Section 7.11.13.7.3). Relevant CSR uses the Parser indirect interface with Table ID 13 (TCAM action RAM).
	ALU 0 Instruction	96	447	Instruction for ALU 0 (see Section 7.11.13.7.2). Relevant CSR uses the Parser indirect interface with Table ID 13 (TCAM action RAM).
	ALU 1 Instruction	96	543	Instruction for ALU 1 (see Section 7.11.13.7.2). Relevant CSR uses the Parser indirect interface with Table ID 13 (TCAM action RAM).
	ALU 2 Instruction	96	639	Instruction for ALU 2 (see Section 7.11.13.7.2). Relevant CSR uses the Parser indirect interface with Table ID 13 (TCAM action RAM).
	Reserved	1	735	Reserved (0).
...			Nx704 + 32	Subsequent entries, 88 bytes each.

7.11.13.9 Protocol Group Table

The Protocol Group table controls hardware expansion of a protocol group into individual protocol and offset pairs in packet metadata. The table contains 192 entries indexed by the Protocol ID output from the Parse Graph. Hardware performs the expansion of the Protocol Group into individual protocols only when the Parse Graph sets the *Is Protocol Group* flag output.

Table 7-252. Package Section Type Numbers for the Protocol Group Tables

Section Type Number	Allowed in Update Package?	Description
57	No	Protocol Group Table for the Rx Parser.
67	No	Protocol Group Table for the Tx Parser.

Each entry in the Protocol Group table defines eight Protocol ID and offset pairs.

The relevant CSR for this table is accessed with GLGEN_ANA_CFG_CTRL.*TABLE_ID* = 12.

The format of this table is shown in Table 7-253.

Table 7-253. Protocol Group Table Section Format

Field		Size (Bits)	Bit Offset	Description
Count		16	0	Number of entries in this section.
Offset		16	16	Offset to the first table address in this section. All entries must be in range 0-191.
Entry 0	Protocol 0	1	32	Polarity of the *Protocol Offset* field. 0b = Positive offset 1b = Negative offset.
		8	33	Protocol ID.
		3	41	Reserved (0).
		10	44	10-bit Protocol Offset.
	Protocol 1	1	54	Polarity of the *Protocol Offset* field. 0b = Positive offset 1b = Negative offset.
		8	55	Protocol ID.
		3	63	Reserved (0).
		10	73	10-bit Protocol Offset.
	...			Entries for Protocols 2-6.
	Protocol 7	1	186	Polarity of the *Protocol Offset* field. 0b = Positive offset 1b = Negative offset.
		8	187	Protocol ID.
		3	195	Reserved (0).
		10	198	10-bit Protocol Offset.
	Reserved	16	208	Reserved(0).
...		192	224	All subsequent entries, 24 bytes each.

7.11.13.10 Marker Group Table

The Marker Group table controls hardware expansion of a marker group into individual marker bits in packet metadata. The table contains 128 entries indexed by the Marker ID output from the Parse Graph. Hardware performs the expansion of the Marker Group into individual marker bits only when the Parse Graph sets the *Is Marker Grou*p flag output.

Table 7-254. Package Section Type Numbers for the Marker Group Tables

Section Type Number	Allowed in Update Package?	Description
72	No	Marker Group Table for the Rx Parser.
73	No	Marker Group Table for the Tx Parser.

Each entry in the Marker Group table defines eight Marker ID values.

The relevant CSR for this table is accessed with GLGEN_ANA_CFG_CTRL.*TABLE_ID* = 16.

The format of this table is shown in Table 7-255.

Table 7-255. Marker Group Table Section Format

Field	Size (Bytes)	Description
Count	2	Number of entries in this section.
Offset	2	Offset to the first table address in this section. All entries must be in range 0-127. Values greater than 127 are reserved.
Entry 0	1	Marker ID 0. All Marker ID values must be in range 0-71, where value 71 represents an invalid (unused) Marker ID.
	1	Marker ID 1.
	1	Marker ID 2.
	1	Marker ID 3.
	1	Marker ID 4.
	1	Marker ID 5.
	1	Marker ID 6.
	1	Marker ID 7.
...	Nx8	All other entries. Each entry occupies a consecutive offset. Maximum number of entries is 128.

7.11.13.11 Parser XLT0 Key Builder Table

The ParserXLT0 Key Builder table specifies the index into the Parser's XLT0 table for each packet.

Hardware builds the key value by extracting bits from the specified metadata field.

Table 7-256. Package Section Type Numbers for the Parser XLT0 Builder Table

Section Type Number	Allowed in Update Package?	Description
53	No	XLT0 Key Builder Table for the Rx Parser.
63	No	XLT0 Key Builder Table for the Tx Parser.

The relevant CSR for this table is GLGEN_ANA_PROFIL_CTRL.

The format of the XLT0 Key Builder Table is shown in Table 7-257.

Table 7-257. Parser XLT0 Key Builder Table Section Format

Field	Size (Bytes)	Description
MDID	1	Specifies the MDID value from the Initialization ID is extracted. Valid range is 0-31. All other values are reserved.
MD Start	1	Start bit for the Initialization ID extraction from the metadata field. Valid range is 0-15. All other values are reserved.
MD Length	1	Length of the Initialization ID extraction from the metadata field. Valid range is 0-31. All other values are reserved.
CD Count	1	The domain key width in the PTYPE marker vector. The only supported value is 3, indicating 8 control domains (72 markers + 8b one-hot control domain bitmap).

7.11.13.12 Package Format of the Parser XLT0 Table

This section specifies the package format for configuring the per-parser XLT0 table. The XLT0 Table provides an intermediate stage during calculation of an Initialization ID. The Parser XLT0 Key Builder Table (see Section 7.11.13.11) builds the key used for XLT0 lookup. The output of the XLT0 table drives the input to the Initialization ID Redirection table (see Section 7.11.13.13).

Each parser uses a distinct XLT0 Table configuration and distinct Package Section Type Numbers to contain the configuration data. Table 7-258 shows the section type enumeration.

Table 7-258. Package Section Type Numbers for the Parser XLT0 Tables

Section Type Number	Allowed in Update Package?	Description
59	No	XLT0 Table for the Rx Parser.
69	No	XLT0 Table for the Tx Parser.

The package format for XLT0 Table configuration is common for all parser as shown in Table 7-259.

Table 7-259. Package Format for XLT0 Table Configuration

Field	Size (Bytes)	Description
Count	2	Number of entries in this section. If the XLT0 table is split into multiple sections, this field describes only the number of entries contained in this section.
Offset	2	Byte offset from the start of the table to the first value in this section. The maximum effective offset for any value is 1023. Effective offsets greater than 1023 are reserved. Relevant CSR: GLGEN_ANA_CFG_CTRL.*TABLE_ADDR*.
Value	1	Value of this XLT0 entry. Valid values are in range (0-15). The value 255 is reserved for internal use by assembler tools before conversion to a valid value. All other values are reserved. The relevant CSR for this table is accessed with GLGEN_ANA_CFG_CTRL.*TABLE_ID* = 26.
...	...	Subsequent values at consecutive offsets.

7.11.13.13 Initialization ID Redirection Table

After the XLT0 table generates the Initialization ID, the Initialization ID Redirection Table can change this ID to a different value. The Redirection Table always maps the input ID to an output ID. Therefore, to allow ID values to pass through unchanged, the host must initialize this table to the trivial identity mapped case (for example, ID 14 maps to ID 14).

Each parser contains its own Initialization ID Redirection Table. Table 7-260 shows the relevant Package Section Type for each block.

Table 7-260. Package Section Type Numbers for Initialization ID Redirection Tables

Section Type Number	Allowed in Update Package?	Description
70	No	Initialization ID Redirection Table for the Rx Parser.
71	No	Initialization ID Redirection Table for the Tx Parser.

The package format for the Initialization ID Redirection Table is common for all parsers as shown in Table 7-261.

Table 7-261. Package Format for Initialization ID Redirection Table

Field	Size (Bytes)	Description
Initialization ID 0 Redirect Value	1	Redirected value for Initialization ID 0. Relevant CSR: GLGEN_ANA_P2P0
Initialization ID 1-14	14 x 1	...
Initialization ID 15 Redirect Value	1	Redirected value for Initialization ID 15. Relevant CSR: GLGEN_ANA_P2P15

7.11.13.14 Metadata Initialization Table

The Metadata Initialization Table provides static values used to initialize packet metadata. The table is indexed using Initialization ID output from the Initialization ID table.

The relevant CSR for this table is accessed with GLGEN_ANA_CFG_CTRL.*TABLE_ID* = 25.

The format of this table is shown in Table 7-262.

Table 7-262. Metadata Initialization Table

Field	Size (Bits)	Bit Offset	Description
Count	16	0	Number of entries in this section.
Offset	16	16	Offset to the first Initialization ID in this section.

Table 7-262. Metadata Initialization Table [continued]

Field		Size (Bits)	Bit Offset	Description
Entry 0	TSR	8	32	TSR - Initial value of the TCAM Search Key register.
	HO	9	40	HO - Initial value of the Header Offset register. Valid range is 0 to 503. All other values are reserved.
	PC	8	49	PC - Initial value of the Program Counter register. Valid range is 0 to 191. All other values are reserved.
	NID	11	57	NID - Initial Parse Graph root node to start package processing. Valid range is 0 to 2047. All other values are reserved.
	CD	3	68	CD - Control Domain ID for this packet. Valid range is 0 to 7. All other values are reserved.
	GPR_A_CTL	1	71	Controls the GPR_A_DATA field. See the description for GPR_A_DATA.
	GPR_A_DATA	16	72	When the GPR_A_CTL is clear, copies a constant value to the GPR specified by GPR_A_ID: GPRS[GPR_A_ID] = GPR_A_DATA When the GPR_A_CTL is set, this field selects the MDID_A value as follows: [4:0] = ID of MDID_A metadata. [8:5] = START extraction start bit for MDID_A [13:9] = LEN extraction length for MDID_A [15:14] = Reserved (0) The specified MDID is copied to GPR_A_ID: GPRS[GPR_A_ID] = MDID_A[LEN+START -1 : START]
	GPR_A_ID	4	88	GPR ID of general purpose register as described above. This field cannot specify the same ID value used in any of the three other *GPR_n_ID* fields.
	GPR_B_CTL	1	92	Same behavior as GPR_A_CTL.
	GPR_B_DATA	16	93	Same behavior as GPR_A_DATA.
	GPR_B_ID	4	109	Same behavior as GPR_A_ID.
	GPR_C_CTL	1	113	Same behavior as GPR_A_CTL.
	GPR_C_DATA	16	114	Same behavior as GPR_A_DATA.
	GPR_C_ID	4	130	Same behavior as GPR_A_ID.
	GPR_D_CTL	1	134	Same behavior as GPR_A_CTL.
	GPR_D_DATA	16	135	Same behavior as GPR_A_DATA.
	GPR_D_ID	4	151	Same behavior as GPR_A_ID.
	Flags	64	155	Initial value for all Flags.
	Reserved	5	219	Reserved(0)
...		Nx192	224	All consecutive entries, 24 bytes each.

7.11.13.15 Last Protocol Table

The Last Protocol Table specifies which Protocol ID values receive special treatment from the parser hardware. The table specifies up to six protocol ID values treated as last protocols. Normally, when the Parse Graph outputs a Protocol ID value, hardware pushes the ID and corresponding Header Offset value onto the Protocol ID stack. However, when this Protocol ID value matches a last protocol, hardware instead records the offset in a side memory. When processing completes, hardware updates the stack with the Protocol ID and last Header Offset value recorded.

Each parser contains its own Last Protocol Table. Table 7-263 shows the relevant Package Section Type for each block.

Table 7-263. Package Section Type Numbers for Last Protocol Tables

Section Type Number	Allowed in Update Package?	Description
74	No	Last Protocol Table for the Rx Parser.
75	No	Last Protocol Table for the Tx Parser.

The package format for the Last Protocol Table is common for all parsers as shown in Table 7-264. The order of entries in table has no effect on functionality.

Table 7-264. Package Format of a Last Protocol Table

Field	Size (Bits)	Bit Offset	Description
Reserved	32	0	Reserved for programming tools use. Ignored by firmware.
Entry 0	1	32	Enable. 1b if this entry is valid, 0b otherwise. Relevant CSR for all fields of this entry: GLGEN_ANA_RX/TX_LAST_PROT_ID[0].
	8	33	
	23	41	Reserved (0).
...	Nx32		All consecutive entries 1-5 formatted as above. Relevant CSR for all fields of this entry: GLGEN_ANA_RX/TX_LAST_PROT_ID[N].

7.11.13.16 Miscellaneous Parser Configuration

The Miscellaneous Parser Configuration section collects together various one-off parser configuration items.

Each parser contains its own Miscellaneous Parser Configuration section. Table 7-265 shows the relevant Package Section Type for each block.

Table 7-265. Package Section Type Numbers for Parser Miscellaneous Configuration

Section Type Number	Allowed in Update Package?	Description
90	No	Miscellaneous configuration for the Rx Parser.
91	No	Miscellaneous configuration for the Tx Parser.

The package format for the Parser Miscellaneous Configuration section is shown in Table 7-266.

Table 7-266. Package Format of the Parser Miscellaneous Configuration Section

Field	Size (Bits)	Bit Offset	Description
PG CAM Hash Key	128	0	The hashing seed value used to calculate the PG CAM entry index as described in Section 7.11.13.2.2. The 45 most significant bits of this value must be 0. A zero value signifies that the hash key is not valid. Firmware ignores this field when the value is zero. Relevant CSR: GLGEN_ANA_PG0_HASHKEY.
No-Match Hash Key	64	128	The hashing seed value used to calculate the PG No-Match CAM entry index as described in Section 7.11.13.2.2. The 13 most significant bits of this value must be 0. A zero value signifies that this hash key is not valid. Firmware ignores this field when the value is zero. Relevant CSR: GLGEN_ANA_NMPG0_HASHKEY.

7.11.13.17 Flags Redirection Table

The Flags Redirection Table specifies if an external flags value is overridden by an internal analyzer flag.

Each parser contains its own Flags Redirection Table. Table 7-267 shows the relevant Package Section Type for each block.

Table 7-267. Package Section Type Numbers for Flags Redirection Tables

Section Type Number	Allowed in Update Package?	Description
97	No	Flags Redirection Table for the Rx Parser.
98	No	Flags Redirection Table for the Tx Parser.

The Package format for the Flags Redirection Table is common for all parsers as shown in Table 7-268. The order of entries in table has no effect on functionality.

Table 7-268. Package Format of a Flags Redirection Table

Field	Size (Bytes)	Offset (Bytes)	Description
Count	2	0	Number of entries in this section.
Offset	2	2	Offset to the first entry (N) in this section. Offset must be in the range 0 to 63.
Entry 0	1	4	The downstream flag identifier, where [0] = When set, the analyzer flag [N] should be exposed on the Flag ID. [6:1] = The internal Flag ID, valid only if the enable bit is set. [7] = Reserved (0) Relevant CSRs: GLGEN_ANA_TX_FLAG_MAP, GLGEN_ANA_FLAG_MAP
---	Nx1 + 4		All subsequent entries in consecutive order, 1 byte each

7.11.14 HIF Block Programming

7.11.14.1 Introduction

The flex descriptor feature allows software entities to program customized receive descriptors.

7.11.14.2 Flex Descriptor Table

The Flex Descriptor table contains descriptor format for a packet for a given descriptor builder profile. Each entry in the Flex Descriptor table defines six Flexible Metadata definitions and up to 18 flags.

Table 7-269. Package Section Type Numbers for Flex Descriptor Table

Section Type Number	Allowed in Update Package?	Description
94	Yes	Rx Flex Descriptors for HIP.

The format of this table is shown in Table 7-270.

Table 7-270. Flex Descriptor Table

Field	Size (Bits)	Bit Offset	Description
Count	16	0	Number of entries in this section.
Offset	16	16	Offset to the first RXDID in this section. The maximum effective offset for any value is 63. Effective offsets greater than 63 are reserved. Values of 0, 1 and 7 are reserved.

Table 7-270. Flex Descriptor Table [continued]

Field		Size (Bits)	Bit Offset	Description
Entry 0	FlexibleMD0	32	32	[7:0] = Protocol ID, MDID Fixed value depending on opcode [17:8] = Offset (for opcode 2), otherwise reserved(0) [29:18] = Reserved (0) [31:30] = Opcode: 00b = Fixed value 01b = Metadata offset 10b = Extraction offset 11b = Protocol offset Relevant CSR: GLFLXP_RXDID_FLX_WRD_0
	FlexibleMD1	32	64	Same format as FlexibleMD0. Relevant CSR: GLFLXP_RXDID_FLX_WRD_1
	...			
	FlexibleMD5	32	192	Same format as FlexibleMD0. Relevant CSR: GLFLXP_RXDID_FLX_WRD_5
	Flag0	32	224	[5:0] The flag index for Flag 0 [7:6] = Reserved (0) [13:8] = The flag index for Flag 1 [15:14] = Reserved (0) [21:16] = The flag index for Flag 2 [23:22] = Reserved (0) [29:24] = The flag index for Flag 3 [31:30] = Reserved (0) Relevant CSR: GLFLXP_RXDID_FLAGS[n, m=0]
	Flag1	32	256	Same format as Flag0. Relevant CSR: GLFLXP_RXDID_FLAGS[n, m=1]
	...			
	Flag4	32	352	[5:0] = The flag index for Flag 16 [7:6] = Reserved (0) [13:8] = The flag index for Flag 17 [31:14] = Reserved (0) Relevant CSR: GLFLXP_RXDID_FLAGS[n, m=4]
...		Nx320 + 32	384	All consecutive entries, 44 bytes each.

7.11.15 RDPU Block Configuration

7.11.15.1 Introduction

The RDPU Block Configuration defines the Receive Data Processing Unit.

7.11.15.2 PTYPE Translation Table

The PTYPE Translation table contains a mapping for E810 10-bit PTYPEs to legacy 8-bit PTYPEs.

Table 7-271. Package Section Type Numbers for PTYPE Redirection Table

Section Type Number	Allowed in Update Package?	Description
95	Yes	PTYPE Translation table for HIF.

Table 7-272. Package Format for PTYPE Redirection Table

Field	Size (Bytes)	Description
Count	2	Number of entries in this section.
Offset	2	Offset to the first PTYPE in this section. Offset must be in range 0 to 1023.
Legacy PTYPE ID	1	Redirected value for the PTYPE specified by the *Offset* field. Relevant CSR: GLFLXP_PTYPE_TRANSLATION
...	Nx1 + 4	Subsequent values in consecutive order.

7.11.15.3 PROTOCOL Table

The PROTOCOL table defines the presence of protocols for which the RDPU verify packet checksum values. If any of the four protocol IDs match for a given protocol table entry, that entry matches.

The PROTOCOL table contains 256 entries corresponding to each possible PTYPE8 value produced by the PTYPE_REMAP table.

Table 7-273. Package Section Type Numbers for the PROTOCOL Table

Section Type Number	Allowed in Update Package?	Description
96	No	PROTOCOL Table for RDPU.

Table 7-274. Package Format for PROTOCOL Table

Field	Size (Bits)	Bit Offset	Description
Count	16	0	Number of entries in this section.
Offset	16	16	Offset to the first entry in this section. Valid values are 0 to 255.

Table 7-274. Package Format for PROTOCOL Table [continued]

Field		Size (Bits)	Bit Offset	Description
Entry 0	Offset Index	3	32	The offset at which the INNER protocol is defined. Valid values are 1-5. A value of 3'b000 indicates not in use. Relevant CSR: GLFLXP_RX_CMD_LX_PROT_IDX
	Reserved	1	34	Reserved (0).
	L4 Offset Index	3	35	The offset at which the L4 Protocol is defined. Valid values are 1-5. A value of 3'000 indicates not in use. Relevant CSR: GLFLXP_RX_CMD_LX_PROT_IDX
	Reserved	1	38	Reserved (0).
	Payload Offset Index	3	39	The offset at which the packet payload is defined. Valid values are 0-5. Relevant CSR: GLFLXP_RX_CMD_LX_PROT_IDX
	Reserved	1	42	Reserved (0)
	L3 Protocol	2	43	The first L3 protocol of the packet. 00b = IPv4 01b = IPv6 10b = Invalid 11b = Other Relevant CSR: GLFLXP_RX_CMD_LX_PROT_IDX
	L4 Protocol	2	45	The last L4 protocol of the packet: 00b = TCP 01b = UDP 10b = SCTP 11b = Other Relevant CSR: GLFLXP_RX_CMD_LX_PROT_IDX
	Reserved	16	48	Reserved (0)
	Protocol Ids for Offset 0	32	64	The protocol IDs for creating offset 0 to RDPU. [7:0] = Protocol ID #0 [15:8] = Protocol ID #1 [23:16] = Protocol ID #2 [31:24] = Protocol ID #3 Relevant CSR: GLFLXP_RX_CMD_PROTIDS[n, m=0]
	Protocol IDs for Offset 1	32	96	Same format as Protocol IDs for Offset 0. Relevant CSR: GLFLXP_RX_CMD_PROTIDS[n, m=1]
	...	...	...	...
	Protocol IDs for Offset 5	32	224	The protocol IDs for creating offset 5 to RDPU. [7:0] = Protocol ID #0 [15:8] = Protocol ID #1 [23:16] = Protocol ID #2 [31:24] = Protocol ID #3 Relevant CSR: GLFLXP_RX_CMD_PROTIDS[n, m=5]
...		Nx224 + 32	256	All consecutive entries, 28 bytes each.

7.12 L2 Packet Processing

7.12.1 CRC Handling

7.12.1.1 Ethernet CRC Insertion

The E810 calculates and inserts the Ethernet CRC for all packets transmitted to the network.

7.12.1.2 Ethernet CRC Stripping

The E810 checks the integrity of the Ethernet CRC and possibly strips it. On packets that are posted to LAN queues, The Ethernet CRC bytes are stripped according to setting of the *CRCStrip* flag in the target LAN queue context

Packets that are routed to PE queue pairs are striped from the Ethernet CRC (with no setting option).

7.12.2 L2 Padding

Transmit packets to the network are padded with zeros to 60 bytes guaranteeing that their length including the CRC is at least 64 bytes. See description of the MAC functionality in Section 3.2.1.3.

Receive packets to the host are padded as well. Packets are padded with zero guaranteeing that they are never shorter than 60 bytes if the following conditions are met:

- Received packet to host memory from the network with no CRC bytes (stripped by the device) and additional stripped fields (like VLAN) that their remaining length is shorter than 60 bytes.
- Loopback packets to host memory (VM to VM) with or without stripped fields (like VLAN) that their remaining length is shorter than 60 bytes.
- Same rules for packets destined to the EMP.

7.12.3 L2 Tag Handling

7.12.3.1 Overview

This section describes the generic mechanism used to handle L2 tags in the E810.

The E810 supports up to eight tags. In general, the text herein applies to each tag independently.

The order these tags are expected in a packet is according to their index in the array described in Table 7-278. Index 7 is the most inner L2 tag (closest to the L3 header) and index 0 is the most outer L2 tag (closest to the MAC Address) as described in Figure 7-36.

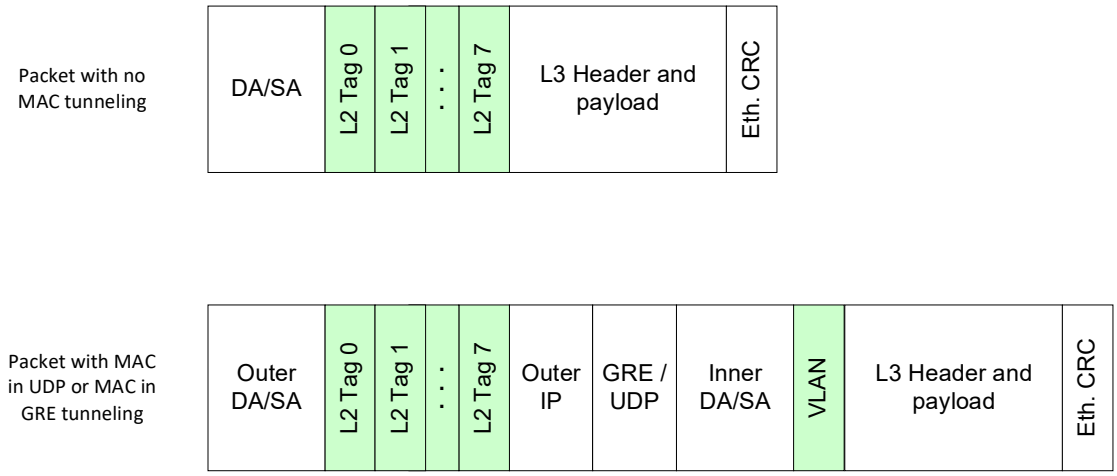


Figure 7-36. L2 Tags Order

Note: The text below refers to offloads provided by the device. A packet received from the network can have any number of tags, and a packet transmitted may have any number of tags added by software provided no offload is required.

Note: When priority bits are mentioned it refers to the 4-MSB-bits of the tag (including the DEI bit).

Note: The L2 tag handling does not use the L2 tag parsing as described in Section A.2, but uses the mechanism described below.

See Section 7.12.3.4 for the programming interface to describe the supported tags and the way to handle them.

7.12.3.2 Transmit Tag Handling

The transmit tag handling has three logical parts as described in the following diagram:

Figure 7-37. Transmit Tag Handling

The software-based tag insertion is described in Section 7.12.3.2.1. The port acceptance rules and port-based tag insertion are described in Section 7.12.3.2.2.

Section 7.12.3.3 describes the receive tag handling and Section 7.12.3.4 describes the software interface used to manipulate the tags.

7.12.3.2.1 Software-Based Tag Insertion Rules

The software can require to send packets with different L2 tag. It can do so either directly through the data for all the tags or using the transmit descriptor as described in Section 10.5.3.1 for up to two tags.

The tags available for the descriptor based insertion are fixed by the device according to the mode of operation.

The type of the tag taken from *L2TAG1* field in this descriptor when *IL2TAG1* field is set is defined by the VSI_L2TAGSTXVALID.*L2TAG1INSERTID* field when VSI_L2TAGSTXVALID.*L2TAG1INSERTID_VALID* is set. If the type of the L2 Tag requires more than two variable bytes, additional bytes are taken from the *L2TAG2* field while the *L2TAG1* is first on the wire. In this case, the *IL2TAG2* flag must be cleared.

The type of the tag taken from *L2TAG2* field in this descriptor when *IL2TAG2* field is set is defined by the VSI_L2TAGSTXVALID.*L2TAG2INSERTID* field when VSI_L2TAGSTXVALID.*L2TAG2INSERTID_VALID* is set.

After the tags are inserted according to the software request, the rules per VSI described in Section 7.12.3.2.2 are applied to decide if the packet can be sent.

7.12.3.2.1.1 Single Tag Handling

Table 7-275 describes the tag insertion in different cases when each tag is identified by a different EtherType:

Table 7-275. Single Tag Handling

Tag in the Data Buffer	Tag in the Tx-Descriptor	Software Requested Action
No	No	Do nothing (send untagged packet).
No	Yes	Insert Tag from the descriptor.
Yes	No	Do Nothing (send packet with the tag from the buffer).
Yes	Yes	This is not a valid configuration and should not be used.

Note: Table 7-275 relates to the configuration of a single tag. The configuration of different tags is independent.

7.12.3.2.1.2 Double Tag Handling

If L2 tags configuration table contains two tags with the same EtherType (for example, two VLANs with EtherType=0x8100) and both tags are enabled, there might be some cases where the hardware might not be able to identify which of the two tags were inserted by the driver directly in the packet.

In this case, if only one tag with that EtherType is seen in the packet and no insertion was requested by the driver or via port-based insertion, it is assumed to be the first of the two tags (in our example, the outer VLAN). Any further action (anti-spoof, UP translation, tag accept policy, and port-based tagging) interpret this tag according to this (so, if anti-spoof, for example, is applied only to inner VLAN, a packet with a single VLAN is treated as a packet with no VLAN for anti-spoof).

If a single tag of these two is present in the buffer sent by the host, but the descriptor requests insertion of one of the tags (for example, an outer VLAN), it is assumed that the tag present in the buffer is the tag for which offload was not requested (in this case, inner VLAN).

After the decision on the identity of each tag, the handling of each tag is as described in Section 7.12.3.2.1.1.

Note: If a tag is inserted by the driver or using port-based insertion, the explanation above is not relevant – the tag is offloaded as required.

7.12.3.2.2 Port (VSI) Based Tag Handling

7.12.3.2.2.1 Accept Rules

This section describes the rules applied per VSI to decide if the software request is acceptable. Table 7-276 summarizes the applied rule according to tag accept, tag insert and the tag presence in the packet for a specific tag.

Table 7-276. Port-Based Tag Handling - Transmit

Tag Accept Mode	Tag Inserted by Driver?	Port-Based Tag	Allowed Driver Behavior[1]	Action
Allow untagged only: VSI_TAR.*ACCEPTTAGGED* = 0 VSI_TAR.*ACCEPTUNTAGGED* = 1	No	No	Yes	Send the packet as is.
	No	Yes	Yes	Tag inserted by the VSI.
	Yes - from descriptor and VLAN ID != 0	Yes	No	VSIs VLAN Tag overrides software while keeping the priority and DEI bits provided by software.
	Yes - in packet	Yes	No	Drop packet.
	Yes	No	No	Drop packet.
	VLAN ID = 0[2]	No	Yes	Send the packet as is.
	VLAN ID = 0[2]	Yes	Yes	Port-based (VSIs) VLAN Tag overrides tag provided by software while keeping the priority and DEI bits provided by software. This mode is supported only if tag is inserted in the descriptor, and only if the insertion is from the first descriptor tag and first VSI tag or from the second descriptor tag and second VSI tag.
Allow tagged and untagged: VSI_TAR.*ACCEPTTAGGED* = 1 VSI_TAR.*ACCEPTUNTAGGED* = 1	X	Yes	Unexpected VSI Setting	Undefined.
	X	No	Yes	Send the packet as is
Allow tagged only: VSI_TAR.*ACCEPTTAGGED* = 1 VSI_TAR.*ACCEPTUNTAGGED* = 0	X	Yes	Unexpected VSI Setting	Undefined.
	No	No	No	Drop packet.
	Yes	No	Yes	Send the packet as is.
Accept nothing VSI_TAR.*ACCEPTTAGGED* = 0 VSI_TAR.*ACCEPTUNTAGGED* = 0	X	X	X	Non-legal configuration.

1. "Unexpected VSI setting" means that the combination of Tag Accept Mode and Port Based Tag should not be requested when creating a VSI.
2. If GL_SWT_L2TAGCTRL.*ISVLAN* is set (can be set only in one tag - inner VLAN).

7.12.3.2.2.2 Port-Based Tag Insertion Mechanism

After the packet sent by the software device driver was accepted, the E810 can add up to three tags based on the VSI that sent the packets. Two of the added tags can have a variable part of up to 16 bits, and one tag can have a variable part of up to 32 bits. The tags that are allowed to be port-based are determined via the VSI_L2TAGSTXVALID.*TIR[012]_INSERT* and VSI_L2TAGSTXVALID.*TIR[012]INSERTID* bits.

The tags for which port-based insertion is done are fixed by the device according to the mode of operation.

For tags up to eight bytes (not including the EtherType), the entire tag can be inserted by the hardware. Either as a request of the software via the descriptor as described in Section 7.12.3.2.1, or as part of a port-based tag insertion.

Each tag can contain the following parts:

- The EtherType
- A fixed part
- A variable part (up to 16 or 32 bits).

The EtherType is taken from the GL_SWT_L2TAGCTRL.*ETHERTYPE* field. The fixed part is taken from the GL_SWT_L2TAGDATA0 and GL_SWT_L2TAGDATA1 register. The GL_SWT_L2TAGCTRL.*LENGTH* field indicates the length of the EtherType payload.

Note: Unused part of the fixed part and the word in which the variable part is inserted should be set to zero.

The variable part is extracted from the descriptor or from the VSI_TIR register. The GL_SWT_L2TAGTXIB.*OFFSET* and the GL_SWT_L2TAGTXIB.*LENGTH* define the location and length of the variable part. Figure 7-38 describes the relationship between the different parameters.

The order of the tags inserted from the port-based tags is according to the order in the bit map. Thus the first bit set uses the value from the VSI_TIR[0] register, the second bit set uses the value from the VSI_TIR[1] register, and the third bit set uses the value from the VSI_TIR[2] register.

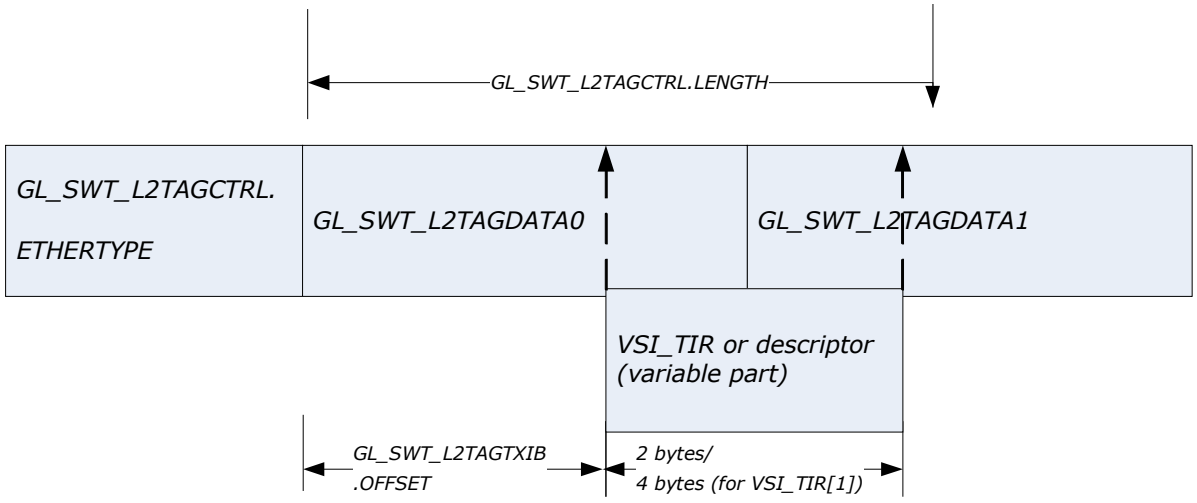


Figure 7-38. Tag Insertion

7.12.3.2.2.3 Queue-Based Tag Insertion

For one of the tags, the E810 can insert different port based tags based on the queue from which the packet was sent. If the ALT_VLAN bit in the transmit queue context is set, the tag is taken from the alternate tag register - VSI_TAIR instead of being taken from VSI_TIR.

This capability is available for each VSI in the tag inserted to the VSI_L2TAGSTXVALID.*TIR0INSERTID* tag.

7.12.3.3 Received Tag Extraction Rules

The tag extraction from receive packets and insertion in the receive descriptor write-back is controlled by the VSI_TSR.*STRIPTAG*, VSI_TSR.*SHOWTAG*, and VSI_TSR.*SHOWPRIONLY* bit fields. The options

supported for each tag are:

- Do nothing.
- Remove this tag from receive packets and do not insert into descriptor.
- Remove this tag from receive packets and insert into descriptor.
- Remove this tag from receive packets and insert only priority bits into descriptor.

Note: The *DEI* bit is considered part of the priority bits and is shown if the priority bits are shown.

Up to two L2 tags can be extracted into the Rx-Descriptor. One of the tags can supply up to 32 bits of data, and the other tag can supply up to 16 bits of data (a total of three 16-bit words max). If more than one word needs to be extracted, 32-byte descriptors need to be used (the *DSize* flag in the receive queue context is set to 32B_Descriptor) otherwise 16-byte descriptors can be used (the *DSize* flag is set to 16B_Descriptor).

The order of the tags extracted to the descriptor is according to the order in the VSI_TSR.*SHOWTAG* bitmap, and according to the *L2TSEL* field in the queue context. If *L2TSEL* bit is set, the first bit set extracts the value to the *L2TAG1* field in the receive descriptor, and the second bit set extracts the value to the *L2TAG2 (2nd)* and *L2TAG2 (1st)* fields in the receive descriptor. If *L2TSEL* bit is cleared, the first bit set extracts the value to the *L2TAG2 (2nd)* and *L2TAG2 (1st)* fields in the receive descriptor, and the second bit set extracts the value to the *L2TAG1* field in the receive descriptor.

Note: The *L2TAG2 (1st)* field is used only if the GL_SWT_L2TAGRXEB.*LENGTH* is 10b or 11b (24 or 32 bits extracted part). Otherwise, the entire tag is stored in L2TAG2 (2nd).

When removal from the packet of a tag is requested, the total L2 tag (determined by the GL_SWT_L2TAGCTRL[7:0].*LENGTH* field), including the EtherType is extracted from the packet.

Further details on the reporting in the descriptor can be found in Section 10.4.2.2.

If the *SHOWIV* bit is set in queue context, the inner VLAN value replaces the tag extracted to L2TAG2.

Table 7-277 describes the content of the tags in the receive descriptor write-back in different cases assuming the first tag is an STag and the second tag is a VLAN.

Table 7-277. Tag Extraction Example

SHOWIV	L2TSEL	Tag #1 (First Bit Set in SHOWTAG (e.g. STag))	TAG #2 (Second Bit Set in SHOWTAG (e.g. VLAN))	Packet			L2TAG1	L2TAG2 (2nd)	L2TAG2 (1st)
				Tag #1 Present	Tag #2 Present	Inner VLAN Present			
0	0	16 bits	16 bits	y	y	X	VLAN	STag	0
				y	n		Not Valid	STag	0
				n	y		VLAN	Not Valid	Not Valid
				n	n		Not Valid	Not Valid	Not Valid
		32 bits	32 bits	y	y		VLAN	Tag#1 LSW	Tag#1 MSW
				y	n		Not Valid	Tag#1 LSW	Tag#1 MSW
				n	y		VLAN	Not Valid	Not Valid
				n	n		Not Valid	Not Valid	Not Valid
		16/32 bits	32 bits	X	X		Not a valid configuration		
	1	16 bits	16 bits	y	y		STag	VLAN	0
				y	n		STag	Not Valid	Not Valid
				n	y		Not Valid	VLAN	0
				n	n		Not Valid	Not Valid	Not Valid
		32 bits	32 bits	y	y		STag	Tag #2 LSW	Tag#2 MSW
				y	n		STag	Not Valid	Not Valid
				n	y		Not Valid	Tag #2 LSW	Tag#2 MSW
				n	n		Not Valid	Not Valid	Not Valid
		16/32 bits	32 bits	X	X		Not a valid configuration		
1	0	16 bits	16/32 bits	y	X	n	Not Valid	STag	0
				n	X	y	Inner VLAN	Not Valid	Not Valid
				y	X	y	Inner VLAN	STag	0
				n	X	n	Not Valid	Not Valid	Not Valid
		32 bits	16/32 bits	y	X	n	Not Valid	Tag#1 LSW	Tag#1 MSW
				n	X	y	Inner VLAN	Not Valid	Not Valid
				y	X	y	Inner VLAN	Tag#1 LSW	Tag#1 MSW
				n	X	n	Not Valid	Not Valid	Not Valid
	1	16 bits	16/32 bits	y	X	y	STag	Inner VLAN	0
				n	X	y	not valid	Inner VLAN	0
				y	X	n	STag	Not Valid	Not Valid
				n	X	n	not valid	Not Valid	Not Valid
		32 bit	16/32 bit	X	X	X	Not a valid configuration		

7.12.3.4 Tag Handling - Programming Interface

This section describes the CSR interface available to control the L2 tags handling of the E810. The actual programming of these capabilities is either through NVM image loading of the device-wide behavior, through port-wide commands such as Set Port Parameters command for the port-wide behavior or through the Add VSI admin command (Section 7.8.12.3.1) for the per-VSI behavior.

The E810 support up to eight types of programmable L2 tags. The types of L2 tags are defined in the NVM and in registers. For each tag the following parameters are defined:

Table 7-278. L2 Tag Control Registers - Global

Register	Field	Description	Section Reference
GL_SWT_L2TAGCT RL [7:0]	ETHERTYPE	The EtherType of the L2 tag.	13.2.2.8.5
	ISVLAN	Is this Tag a VLAN tag. This information is used to define if priority tagging should be supported for this tag.	
	INNERUP	If this bit is set, the UP remapping is done on this field. If this bit is set, *ISVLAN* should also be set. If set in multiple tags, the Inner UP is taken from the first tag with this bit set in the packet.	
	OUTERUP	If this bit is set, this is the tag on which the inner to outer UP remapping is applied. Should be set only in one tag.	
	LENGTH	The length of the L2 tag (not including the EtherType). The length can be 2,4,6 or 8 bytes.	
	HAS_UP	Defines if this tag includes UP bits that should be used for UP to TC translation. The first such tag found in the packet is used for UP to TC translation	
	ISNSH	Identifies the tags that contains a NSH EtherType.	
	ISMPLS	Identifies the tags that contains a MPLS EtherType.	
PRT_L2TAGSEN	NONLAST_TAG	Indicates that there are more tags with the same EtherType expected.	13.2.2.26.1
GL_SWT_L2TAGTXIB [7:0]	OFFSET	Describes the offset in the header to which the variable data should be inserted (can be up to eight bytes)	13.2.2.8.3
GL_SWT_L2TAGDA TA0[7:0] GL_SWT_L2TAGDA TA1[7:0]	L2TAGDATA	The fixed part to insert in transmit packets	13.2.2.8.1 13.2.2.8.2

Each port defines which tags to expect in packets sent and received through this port using the GL_L2TAGSEN_<0/1/2/3>_<0/1> registers according to the following table per mode as defined by the GLGEN_MAC_LINK_TOPO register:

Table 7-279. Number of Ports to L2 Tags Enable Mapping

Port Number	Number of Ports		
	2	4	8
0	GL_L2TAGSEN_0_0 and GL_L2TAGSEN_1_0[1]	GL_L2TAGSEN_0_0	GL_L2TAGSEN_0_0
1	GL_L2TAGSEN_2_0 and GL_L2TAGSEN_3_0[1]	GL_L2TAGSEN_2_0	GL_L2TAGSEN_2_0
2		GL_L2TAGSEN_1_0	GL_L2TAGSEN_1_0
3		GL_L2TAGSEN_3_0	GL_L2TAGSEN_3_0
4			GL_L2TAGSEN_0_1
5			GL_L2TAGSEN_2_1
6			GL_L2TAGSEN_1_1
7			GL_L2TAGSEN_3_1

1. The two registers should have the same value.

The per-port PRT_TDPUL2TAGSEN defines which packets are expected in packets transmitted from this port.

Each VSI defines which of the eight L2 tags are offloaded for its Tx and Rx traffic using the VSI_L2TAGSTXVALID and VSI_TSR registers. These registers define which receive tags and transmit tags to handle.

Each VSI can be configured with a different behavior for each of the tags. The possible behaviors relate to the type of tags the driver is allowed to insert in transmit packets, the type of tags inserted by the switch, the tags removed from received packets, and the tags posted to the receive descriptor write-back.

Table 7-280 describes the registers fields used to set the expected behavior for each tag.

Table 7-280. L2 Tag Control Registers - per VSI

Register	Field	Description	Section Reference
VSI_L2TAGSTXVALID	L2TAG[12]INSERTID[1] L2TAG[12]INSERTID_VALID	Defines the tags for which descriptor-based insertion is supported. The ID is based on the L2 tags mapping described in Table 7-281.	13.2.2.12.6
	TIR[012]INSERTID TIR[012]_INSERT	Defines the tags for which port-based insertion is supported.	
VSI_TAR[2]	ACCEPTTAGGED[n]	A bitmap describing if a packet with tag N is accepted.	13.2.2.12.5
	ACCEPTUNTAGGED[n]	A bitmap describing if a packet without tag N is accepted.[3,4]	
VSI_TIR_n (n = 0..2)	PORT_TAG_ID	The tag to insert.	
VSI_TAIR	PORT_TAG_ID	The alternate tag that can be used instead of VSI_TIR[0]	

Table 7-280. L2 Tag Control Registers - per VSI [continued]

Register	Field	Description	Section Reference
VSI_TSR	STRIPTAG[n] (n = 0..9)	Defines if the tag should be extracted from the packet.	13.2.2.12.10
	SHOWTAG[n] (n = 0..9)	Defines which of the tags should be extracted to the descriptor. Valid only if corresponding bit in *STRIPTAG* is set. The *SHOWPRIONLY* field defines which part of the tag to extract to the descriptor. At most, two of these bits should be set. If more than two bits are set, only the first two are considered.	
	SHOWPRIONLY[n] (n = 0..9)	A per-tag bitmap defining which par of the tags to extract to the descriptor. If set, only the priority bits are extracted. Otherwise, the entire tag is used. Relevant only if the corresponding bit in *SHOWTAG* is set.	

1. To allow insertion of priority bits from a descriptor and the VLAN tag value from the hardware as described in Table 7-275, the same ID should be used for L2TAG1INSERTID and TIR0INSERTID or L2TAG2INSERTID and TIR1INSERTID.
2. The tag number in this register relates to the 10 tags defined in Section 7.12.3.5.
3. If GL_SWT_L2TAGCTRL.*ISVLAN* is set, admits also priority tagged packets (VLAN tag = 0).
4. This bit should be set for all the tags not expected in the packet.

7.12.3.5 L2 Tags Default Configuration

This section describes the value to set in the different registers to implement the E810 POR as loaded from NVM.

The following L2 tags are currently defined in the default configuration:

Table 7-281. L2 Tags

Tag	Tag ID (in Table)	Tag ID in VSI_TAR and VSI_TSR bitmaps
STag	0	2
ITag (MAC in MAC)	1	3
Outer VLAN (0x8100)	2	4
Outer VLAN (0x9100)	3	5
VLAN	4	6
NSHoE	5	7
MPLS 1	6	8
MPLS 2	7	9

Table 7-282 describes the configuration for each of the tags:

Table 7-282. L2 Headers Support

Register/Tag	STag	ITag (MAC in MAC)	Outer VLAN 1	Outer VLAN 2	VLAN	NSHoE	MPLS 1	MPLS 2
Index	0	1	2	3	4	5	6	7
Global Configuration								
GL_SWT_L2TAGCTRL. *ETHERTYPE*	0x88A8	0x88E7	0x8100	0x9100	0x8100	0x894F	0x8847	0x8848
GL_SWT_L2TAGCTRL.*ISVLAN*[1]	0	0	0	0	1	0	0	0
PRT_L2TAGSEN.*NON_LAST_TAG*	0	0	1	0	0	0	0	0
GL_SWT_L2TAGCTRL.*INNERUP*	0	0	0	0	1	1	0	0
GL_SWT_L2TAGCTRL.*OUTERUP*	1	0	1	1	0	0	0	0
GL_SWT_L2TAGCTRL.*LONG*	0	0	0	0	0	0	0	0
GL_SWT_L2TAGCTRL.*HAS_UP*	1	0	1	1	1	1	0	0
GL_SWT_L2TAGCTRL.*ISMPLS*	0	0	0	0	0	0	0	1
GL_SWT_L2TAGCTRL.*ISNSH*	0	0	0	0	0	0	1	0
GL_SWT_L2TAGCTRL.*LENGTH*	2	16	2	2	2	2	36	0
GL_SWT_L2TAGTXIB.*OFFSET*	0	N/A	0	0	0	N/A[2]	N/A	N/A
GL_SWT_L2TAGTXIB.*LENGTH*[3]	01b	N/A	01b	01b	01b	N/A	N/A	N/A
GL_SWT_L2TAGRXEB.*OFFSET*	0	N/A	0	0	0	N/A	N/A	N/A
GL_SWT_L2TAGRXEB.*LENGTH*[3]	01b	N/A	01b	01b	01b	N/A	N/A	N/A
GL_SWT_L2TAGDATA0, GL_SWT_L2TAGDATA1	All zeros							
Per Port Configuration (Controlled by Set Port Parameters command)								
PRT_L2TAGSEN.*ENABLE* and GL_L2TAGSEN_m_n (m= 0..3, n= 0..1)	1	1	0/1[4]	1	1	1	1	1
Per VSI Configuration (Controlled by Add VSI command)								
VSI_TSR.*STRIPTAG*	0/1[5,6]	0	0/1[5]	0/1[5]	0/1[7]	0	0[8]	0[8]
VSI_TSR.*SHOWTAG*	0/1[6,9]	0	0/1[9]	0/1[9]	0/1[10]	0	0[8]	0[8]
VSI_TSR.*SHOWPRIONLY*	0	0	0	0	0/1[11]	0	0[8]	0[8]
VSI_TAR.*ACCEPTTAGGED*	0/1[12]	1	0/1[12]	0/1[12]	0/1[13]	1	1[14]	1[14]
VSI_TAR.*ACCEPTUNTAGGED*	1	1	1	1	1[15]	1	1[14]	1[14]

1. The *ISVLAN* must be set on one Tag and only on one Tag (the inner VLAN).
2. Insertion and extraction of tag ID is not supported for this tag.
3. 00b = 8 bits, 01b = 16 bits, 10b = 24 bits, 11b = 32 bits.
4. Should be set if port expects double VLAN packets.
5. Set for the used outer tag if outer tag extract mode is set to Extract tag and do not insert in descriptor or Extract tag from packet and expose in descriptor.
6. If STag is used as part of a B-VLAN-ITag combo, bit should be cleared.
7. Set if *VLAN and UP Expose Mode (Rx)* is not set, do nothing (leave VLAN in packet).
8. No L2 tag extraction support for MPLS tags.
9. Set for the used outer tag if outer tag extract mode is set to Extract tag from packet and expose in descriptor.
10. Set if *VLAN and UP Expose Mode (Rx)* is set to Show VLAN, DEI and UP in descriptor.
11. Set if *VLAN and UP Expose Mode (Rx)* is set to Hide VLAN, show UP and DEI (VLAN ID = 0).
12. Set for the used outer tag if *Accept Tag from Host* is set. Set for all other tags.
13. Set if *VLAN Driver Insertion Mode* is set to Admit.1Q tagged only or Allow all packets.
14. No check of the insertion rules of MPLS headers.
15. Set if *VLAN Driver Insertion Mode* is set to Admit untagged/Priority tagged only or Allow all packets.

7.12.4 VLAN Handling

This section describes the handling of IEEE 802.1Q VLAN tags based on the features described above.

There can be up to two VLAN tags in the outer header of a packet identified as VLAN (tag index 5) and outer VLAN (tag index 3 or 4). In addition, the tunneled header can also include a VLAN. The handling of the tunneled VLAN is described in Section 7.12.4.3.

Each port can be set to expect packets with outer VLAN using the Set Port Parameters admin command (Section 7.8.12.2.2). When enabled, a packet with a single VLAN is treated as a packet with outer VLAN only. Otherwise, a single VLAN in a packet is treated as inner VLAN.

In any case, the outer VLAN is not part of the forwarding decision and should be handled by the software device driver or by the BMC both in transmit and receive. There is no offload of outer VLAN insertion or extraction.

The following sub-sections describe the handling of the inner VLAN.

In UDP tunnels, an internal VLAN can also be present, as described in Section A.5.5. The handling of this VLAN is described in Section 7.12.4.3.

UP translation in inner VLAN is described in Section 7.12.6.2.

7.12.4.1 Transmit Flow

This section describes the handling of VLAN as part of the flow of packets sent by the host, the BMC, or the EMP.

7.12.4.1.1 Tag Insertion

A VLAN tag can be inserted to the packets in three ways:

- As part of the packet buffer.
- As part of the transmit descriptor in the *L2TAG1* field if the *IL2TAG1* field is set.
- By the device from the VSI context.

The two first options are enabled if the VSI is allowed to add a VLAN tag by the *VLAN Driver Insertion Mode* in the Add VSI command (Section 7.8.12.3.1). If a packet is sent with a VLAN tag from a VSI that is not allowed to add a tag, it is dropped.

Note: If the *IL2TAG_IL2H* field in the transmit context descriptor is set, the *L2TAG1* represents the inner VLAN and regular VLAN insertion from the descriptor is not available. See Section 7.12.4.3.1 for details on the inner tag insertion. Inner tag insertion is allowed irrespective of the VLAN configuration in the Add VSI command.

The third option is enabled by setting the *Insert PVID* in the same command. The tag to insert is defined in the *PVID + Default UP* field of the command.

Note: Setting both *Insert PVID* and *VLAN Driver Insertion Mode* to allow software to insert VLAN is not allowed.

Note: It is expected that BMC traffic arrives with the right VLAN tagging.

7.12.4.1.2 VLAN Anti-Spoofing

After the packet is VLAN tagged by one of the methods above, if the *Enable VLAN Anti-Spoof* bit in the Add VSI command is set, it is compared to the ingress VLAN list and dropped if the inserted VLAN is not in the list.

7.12.4.1.3 VLAN Filtering

If the packet passed the previous stage, the VLAN tag is used as part of the forwarding process. It is compared as part of the {MAC, VLAN} filters added by the Add Switch Rules admin command (Section 7.8.12.6.1) to determine if the packet should be sent to a local address or should be sent to the network. It is also compared to the PRT_MNG_MAVTV filters as part of the manageability filtering as described in Section 12.4 to define if it should be sent to the BMC.

7.12.4.2 Receive Flow

This section describes the handling of VLAN as part of the flow of packets received by the host, the BMC, or the EMP.

7.12.4.2.1 VLAN Filtering

When a packet is received from the network (or from the host) the VLAN tag is used as part of the forwarding process. It is compared as part of the {MAC, VLAN} filters and to the VLAN egress filtering both set using the Add Switch Rules admin command (Section 7.8.12.6.1) to determine if the packet should be sent to a local VSI. It is also compared to the PRT_MNG_MAVTV filters as part of the manageability filtering as described in Section 12.4 to define if it should be sent to the BMC.

7.12.4.2.2 VLAN Extraction

Before a packet is stored in the host memory, the VLAN tag can be stripped and optionally stored in the receive descriptor. The action done is defined per VSI in the *VLAN and UP Expose Mode (Rx)* field in the Add VSI command. The possible actions are:

- Show VLAN and UP in descriptor (legacy behavior).
- Hide VLAN show UP in descriptor (VLAN ID exposed as 0).
- Hide VLAN and UP.
- Do nothing (leave VLAN in packet).

If the VLAN or the UP are exposed in the descriptor, it shows in the *L2TAG1* field and the *L2TAG1P* flag is set. If the *L2TSEL* bit is set, VLAN tag is extracted to the *L2TAG2 (1st)* field in the receive descriptor instead. In this case, 32-byte descriptors must be used (*DSize* field in the receive queue context must be set).

Note: When a packet is sent to the BMC the VLAN is kept in the packet.

7.12.4.3 VLAN in Tunnel Packets

In MAC-in-UDP and MAC-in-GRE encapsulations, a VLAN within the tunneled MAC header can also be present, as described in Section A.5.5.

7.12.4.3.1 Insertion of Tunneled VLAN from Descriptor

The insertion of this tag is controlled via the *IL2TAG_IL2H* field in the transmit descriptor. When set, the *L2TAG2* field in the context descriptor contains the tunneled VLAN. In this case, the values of VSI_L2TAGSTXVALID.*L2TAG2INSERTID* and VSI_L2TAGSTXVALID.*L2TAG2INSERTID_VALID* fields are ignored.

This mode is enabled irrespective of the VLAN configuration in the Add VSI command (Section 7.8.12.3.1).

Note: Insertion of inner VLAN should not be requested when IPsec encryption is requested.

Inner VLAN insertion is based on the inner IP offset in tunneled packets, and is assumed to be inserted before the inner IP header. For packets for which the insertion should be done elsewhere (for example, if an MPLS header is present between the inner VLAN and the inner IP), the VLAN cannot be inserted by the device.

7.12.4.3.2 Extraction of Tunneled VLAN to Descriptor

The *SHOWIV* field in the receive queue context controls the extraction of an internal VLAN to the receive descriptor. If set, the tunneled VLAN is inserted in *L2TAG2 (2nd)* field of the receive descriptor write-back. In this case, the second tag selected according to Section 7.12.3.3 is ignored. If the *L2TSEL* bit is cleared, the inner VLAN tag is extracted to the *L2TAG1* field in the receive descriptor instead. When the *SHOWIV* field is set and the *L2TSEL* bit is set, 32-byte descriptors must be used (*DSize* field in the receive queue context must be set).

7.12.4.3.3 Tunneled VLAN in Pass-Through Traffic

Tunneled VLAN is not offloaded (not inserted nor extracted) for BMC pass-through traffic.

7.12.4.4 Port-Based VLAN

The E810 supports port-based VLAN feature by allowing any VSI to specify the default VLAN that it belongs to. The port-based VLAN association is done when a packet received on the VSI is untagged or priority tagged and protocol VLAN association is not done. Port-based VLANs map packets received on a given VSI with the corresponding Port based VLAN identifier called PVID. The PVID list should be programmed with the Port VLAN IDs for each VSI.

The port VLAN list is provided which is part of the VSI context. Table 7-283 describes the port VLAN parameters in the Port VLAN list and the matching parameters in the Add VSI command.

Table 7-283. Port VLAN List and VSI Parameters

Port-Based VLAN List	Add VSI Parameter	Notes
VSI number	VSI Number	Returned by firmware in response.
PVID	PVID + Default UP	
Default UP		
Admit.1Q tagged only	VLAN Driver Insertion Mode	
Admit untagged/Priority tagged only		
Admit all		
Ingress VLAN check enable	Enable VLAN Anti-Spoof	
Insert PVID	Insert PVID	

Table 7-283. Port VLAN List and VSI Parameters [continued]

Port-Based VLAN List	Add VSI Parameter	Notes
Expose VID and UP of received packets	VLAN and UP Expose Mode (Rx)	
Expose UP only of received packets		
Do not expose VID or UP of received packets		
Ingress UP translation table	Ingress UP Translation Table	See Section 7.12.6.1
Egress UP translation table	Egress UP Translation Table	See Section 7.12.6.2

The switch should insert the PVID to the packet if the *Insert PVID* parameter is set for the VSI and replace the UP bits according to the algorithm described in Section 7.12.6.1.

Once the VLAN ID association is made, the packet forwarding is performed using the {VLAN, MAC} forwarding table and VLAN membership table.

7.12.5 Outer Tag Handling

This section describes the handling of outer VLAN tags (IEEE 802.1Qbg STags or 0x8100 /0x9100 VLAN tags) based on the features described above.

There can be up to one outer tag in a packet (tag index 1/3/4/). The sections below describes the handling of the STag.

Note: The outer tag EtherType can be either 0x88A8, 0x9100, or 0x8100 and can be different per VSI according to the *Outer Tag Type* defined in the Add VSI command.

7.12.5.1 Transmit Flow

This section describes the handling of outer tag as part of the flow of packets sent by the host, the BMC, or the EMP.

There is no anti-spoofing capability for outer tags.

7.12.5.1.1 Tag Insertion

An outer tag can be inserted to the packets in three ways:

- As part of the packet buffer.
- As part of the transmit descriptor in the *L2TAG2* field if the *IL2TAG2* field is set.
- By the device from the VSI context.

The two first option are enabled if the VSI is allowed to add an outer tag by the clearing the *Outer Tag Insert Enable* field and setting the *Accept Tag from Host* field in the Add VSI command (Section 7.8.12.3.1). If a packet is sent with an outer tag from a VSI not allowed to add a tag, it is dropped.

The third option is enabled by setting the *Outer Tag Insert Enable* in the same command. The tag to insert is defined in the *Outer Tag* field of the command.

Note: Setting both *Outer Tag Insert Enable* and *Accept Tag from Host* to allow software to insert VLAN is not allowed.

Note: As an outer tag of 0x8100 can be confused with an inner tag of the same value, it should be enabled in the Set Port Parameters command only if expected on all packets in this port. In this case, other outer tags (0x9100 or 0x88A8) can not be inserted by the hardware.

7.12.5.2 Receive Flow

This section describes the handling of STag as part of the flow of packets received by the host, the BMC or the EMP.

7.12.5.2.1 STag Extraction

Before a packet is stored in the host memory, the STag can be stripped and optionally stored in the receive descriptor. The action done is defined per VSI in the *STag Extract Mode* field in the Add VSI command. The possible actions are:

- Show STag and UP in descriptor (legacy behavior) — Used for cascaded Port Virtualizer with offload.
- Hide STag and UP.
- Do nothing (leave STag in packet) — Used for cascaded Port Virtualizer without offload.

If the STag is exposed in the descriptor, it shows in the *L2TAG2* field and the *L2TAG2P* flag is be set. In this case, 32-bytes descriptors must be used (*DSize* field in he receive queue context must be set).

Note: If the *SHOWIV* field is set in the queue context, the *L2TAG2* field is allocated for the inner VLAN of tunnel packet. In this case, the STag is never inserted in the descriptor.

7.12.6 User Priority Bits (802.1p) Handling

The UP bits are used to differentiate between multiple classes of traffic. The E810 supports multiple use cases related to the handling of the UP bits:

- An OS that is not DCB/traffic type aware and uses a single TCID.
- An OS that is traffic type aware, but not DCB aware. It can distribute the traffic to different queues, but cannot tag them with the right UP. This case refers to LAN and iWARP flows. The OS is not aware, but the driver knows the difference and hence can assign the different types of traffic to different flows. There is no DCBX agent and hence driver does not know the UP or TC for flows.
- An untrusted OS that can set UP bits of TCs it is not allowed to use.
- An OS that can use a single queue per TCID, but would like to use more UPs to gain from the differentiated QoS in the network.

The following sub-sections describe the handling of UP bits in transmit and receive to support the above use cases.

See Section 8.2.1.3.1 for details of UP handling in ports where the TC is DSCP-based.

7.12.6.1 Transmit Functionality

Each transmit packet is assigned a UP, either by the driver or as part of the port VLAN table as described in Section 7.12.4.4.

This UP is translated using two different translation vectors:

- The first vector is specific to a VSI and is part of the port VLAN table (Transmit UP translation table - VSI_TUPR). This vector reflects the mapping of the user priorities as seen by the operating system to the user priorities as seen by the network.
- The second vector is specific to a TCID in a physical port. It translates the UP received from the previous translation to a UP matching the TCID to which the packet is associated. These vectors can be configured through the PRT_TCTUPR registers.

Note: There is no drop of packets due to a wrong 802.1p tagging. If a wrong tag is requested, the tag is replaced. Thus, this functionality supports both UP anti-spoofing and port-based UP.

Untagged packets are kept as is. No translation is done on them.

Figure 7-39 shows the algorithm:

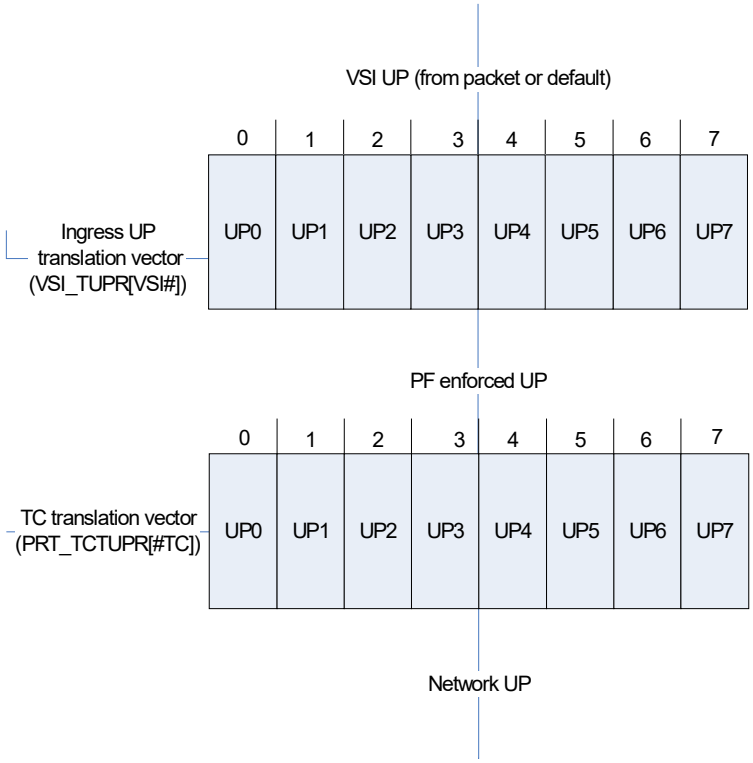


Figure 7-39. Example: Non-VLAN Aware, Non-DCB Aware OS

Assume a VF that queues LAN traffic and iSCSI traffic to different queues, but is not VLAN-aware. In this case, the packet is sent by the driver with no VLAN, and the default UP set in the port-based VLAN (for example, 0) is the initial UP of the packet. In this case, the Ingress UP translation vector (VSI_TUPR) is not relevant and might be, for example, {0,1,2,3,4,5,6,7} or {0,0,0,0,0,0,0,0}. The TC translation vector (PRT_TCTUPR) for LAN might be {0,1,2,0,0,0,0,0} and the TC table for storage might be {4,4,4,4,5,4,4,4}. So a LAN packet goes out with a UP of 0, and a storage packet goes out with a UP of 4.

Example: Non DCBX Tagging OS

Assume a VF that tags LAN high-priority traffic with a UP of 4, LAN low-priority traffic with a UP of 3, and the storage traffic as UP 7. In this case, the VF translation table might be {0,1,2,0,2,4,4} and the TC tables as above. The low-priority LAN goes out with a UP of 0, the high-priority LAN with a UP of 2 and, storage with a UP of 4.

The tag on which the transmit UP translation is done is identified by setting the *INNERUP* bit in the matching GL_SWT_L2TAGCTRL register.

The default mapping of both tables is identity mapping (that is, mapping a UP to itself). If UP translation is not needed, these mapping should not be changed.

7.12.6.1.1 Transmit Outer Tag User Priority

The transmit outer tag user priority can be handled in three different ways:

1. If the outer tag is received in the packet or the descriptor (for example, cascaded S-comp), the UP should be part of the packet sent by the driver or inserted from the descriptor.
2. If the outer tag is inserted by the hardware, the UP can be either:
   a. The UP defined as part of the inserted tag in the VSI_TIR register.
   b. A translation of the regular VLAN UP as defined in the VSI_TUPIOM register.

If the outer tag is defined as *OUTERUP* in the GL_SWT_L2TAGCTRL register, the UP is based on the first tag for which the *INNERUP* field is set in the GL_SWT_L2TAGCTRL register (usually the VLAN tag) (option 2.b). Otherwise, it is based on the default (option 2.a).

In case there is more than one tag in the packet with *OUTERUP* set, inner-to-outer UP translation should be performed to the most outer tag with *OUTERUP* bit set.

The tag to which translation is applied can be either STag or external VLAN.

Figure 7-40 describes the translation process.

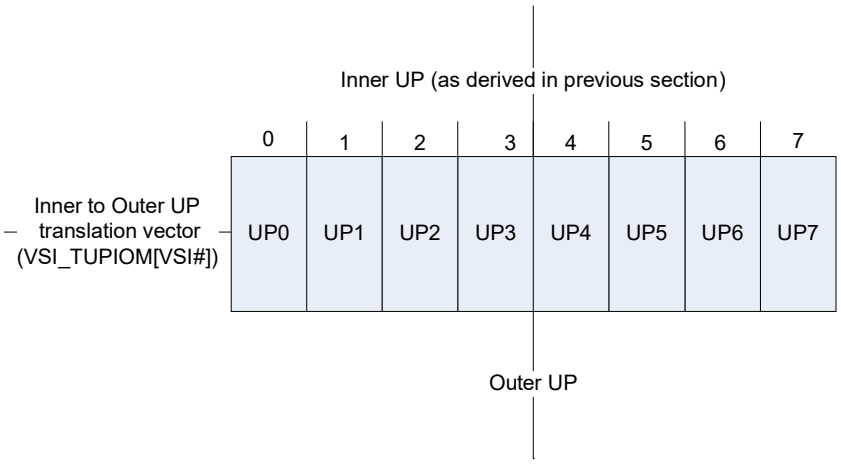


Figure 7-40. Tx Inner-to-Outer UP Translation

7.12.6.2 Receive Functionality

7.12.6.2.1 VLAN UP Translation

The priority bits of inner VLAN tag in received packets may be remapped using the Ingress UP translation table in the Add VSI command. This means the OS might get packets with 802.1p priority bits reflecting the local configuration and not the link configuration. By default (Ingress UP translation section is not valid) the translation is one-to-one, meaning that the received UP is not translated.

7.12.6.2.2 VLAN UP Exposure to Device Driver

The exposure of the received UP to operating systems is defined by the awareness of the OS controlling the VSI as follows:

- For a monolithic OS, VMM, or for a guest OS that is VLAN-aware, the UP is exposed as part of the VLAN as requested by driver (either in the packet or in the descriptor).
- If the guest OS is DCB-aware but not VLAN-aware, it gets the priority bits as part of the receive descriptor in the VLAN tag field. The VLAN ID is zero, but the priority bits are valid.
- A guest OS that is not VLAN-aware and not DC-aware does not get the user priority bits (UP) at all.

The configuration of the UP removal is part of the port VLAN configuration described in Section 7.12.4.4, and in the Add VSI command VLAN handling section (Section 7.8.12.3.1).

NOTE: ***This page intentionally left blank.***

Chapter 8 Quality of Service (QoS)

This chapter assumes the reader is familiar with the following specifications:

- IEEE P802.1Qbb-2011 — a.k.a. Priority-based Flow Control (PFC) specification.
- IEEE P802.1Qaz-2011 — a.k.a. Enhanced Transmission Selection for Bandwidth Sharing Between Traffic Classes (ETS) specification.
  - DCB Center Bridging Exchange Protocol (DCBX) spec refers to Clause 38 in ETS specification.
  - DCB Capability Exchange Protocol Base Specification.
- IEEE Std 802.1AB-2009 — a.k.a. Link Layer Discovery Protocol (LLDP) specification.
- RFC 2474—Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers.

8.1 E810 Usage Models: Number of Ports and Number of Congestion Domains

The E810 supports various port configuration and usage models.

- The E810 is integrated to the Network Acceleration Complex (NAC) SoC. It is targeted for COMMS, Cloud, and Enterprise markets. The E810 operates as a NIC configured with up to eight ports. The E810 is directly connected to PHYs with eight ONPI interfaces, each directly connected to one PHY for both Tx and Rx.
- **Congestion Notification:** Congestion notifications from the network (Link Flow Control or Priority-based flow control) is immediately forwarded to the E810 like other traffic.
- the E810 supports three main ports settings. Each Port is associated with a separate ONPI connectivity to the Network.
  - Two ports setting:
    - Each port can be independently run at 100/50/25/10/1 GbE, 100 Mb/s, or disabled.
    - Eight TCs (Congestion Domains) are supported per port in both Tx and Rx sides.
  - Four ports setting:
    - Each port can be independently run at 25/10/1 GbE, 100 Mb/s, or disabled.
    - Eight TCs (Congestion Domains) are supported per port in both Tx and Rx sides. Total 32 Congestion Domain (CGDs).
  - Eight ports setting (Device mode is set to this mode whenever more than four ports are available):
    - Each port can be independently run at 10/1 GbE, 100 Mb/s, or disabled.
    - Four TCs (Congestion Domains) are supported per port in both Tx and Rx sides. Total 32 Congestion Domain (CGDs).
    - Tx-Scheduler configuration in this usage model is detailed in Section 8.3.3.

    Note: This port setting supports two more ports sub settings:
    - Six ports setting: supports 2x25 GbE + 4x10 GbE ports setting.

- Seven ports setting: supports 1x25 GbE + 6x10 GbE ports setting.

From QoS perspective, the above modes are sub setting of eight ports mode and four TCs (Congestion Domains) are supported per port. Some CGD are not used in those modes.

— LLDP/DCBX: LLDP and DCBX agent runs in EMP firmware by default. Software can acquire the LLDP ownership.

8.2 E810 QoS and DCB Support

8.2.1 Receive Path QoS

This section discusses the QoS approach in the E810. QoS in receive includes DCB and DCBX support plus Rx-Pipe balancing to avoid starvations between Rx-Pipe units and between different types of service packets in the Rx processing pipeline. This section includes:

- Rx-Pipe overview of QoS aspect on the Rx-Pipe:
  - — Describes the potential congestion domains, Arbitration and drop points, and Priority support.
- Receive QoS, Algorithms, and Configuration:
  - — Rx QoS units Overview:
    - Incoming packet classification to CGD
    - Rx-Pipe - arbiters
    - Rx-Pipe - “pipe monitors”
  - — Rx QoS blocks, Implementation details, and Programming:
    - Incoming packet classification to CGD
    - Rx-Pipe - arbiters
    - Rx-Pipe - “pipe monitors”
- XOFF/XON methods differences in Device operation modes.
- Crosstalk prevention.

8.2.1.1 Rx-Pipe Congestions, Arbitration, and Priority Support - Overview

The E810’s Rx-Pipe processes incoming packets from the network as well as loopback traffic. Under normal configuration, the PCIe bandwidth provides high enough bandwidth for total incoming traffic and its control data (like the descriptors write-back).

The Rx-Pipe might be congested as a result of one or more of the below conditions:

- A thinner PCIe connectivity limits the Host interface.
- A burst of high-rate small packets, congest the Rx packet processing pipeline.
- A burst of RDMA packets, overflows the processing power of the PE (PE packet processing rate is lower than the LAN packet processing rate).

- Dual 100 GbE mode: The E810 can be configured to work in 2x100 GbE setting. In that topology, the receive Packet Buffer can be congested by receiving from the ports more packets than can be fed to the receive pipe (higher rate than Rx-Pipe processing power).
- A combination of Loopback and Mirroring Traffic in parallel to high-rate incoming packets from the network.
- Rx locality rules: The E810 performance commitment counts on a level of locality. The background behind the locality restriction for Rx performance is that some of the stages of Rx processing are dependent on fetching data from the Host memory, part of this data can be a Rx-Descriptor per each packet, other type of data is a context type like filters. To eliminate the need for the Host read operations per packet, the E810 manages an internal cache for all those data types. The E810 can benefit from those cache only under some locality assumptions. When the incoming traffic conditions do not follow those locality conditions, the Rx processing rate of the E810 suffers from some performance degradation, and part of the Rx-Pipe is congested.
- Unexpected collision numbers in Post Quad Hash (discussed later in this section) can also cause congestion.

When the Rx-Pipe is congested as a result of any of the above, internal processing queues are starting to build up. The Rx-Pipe is organized in per CGD queues/FIFOs. Packet reordering is only allowed between CGDs.

As detailed above, in the Rx side, the E810 can be configured to work in one of three main port topologies:

- Four ports with up to eight TCs
- Eight ports with up to four TCs

The Rx-Pipe is organized in up to 32 queues, one per congestion domain (CGD). The processing of packets belonging to a CGD is done in order.

Association of incoming packets with a CGD is done based on the port's QoS configuration and the packet's Class of Service (CoS) as detailed in Section 8.2.1.3.1.

Some stages of the pipe might need higher processing power and therefore two units are placed to meet the required performance. Those pipe stages operate internal re-ordering mechanism to keep the overall pipe ordering.

When internal queues are built up, the processing order in Rx-Pipe must implement an arbitration between the per CGD queue/FIFOs.

8.2.1.1.1 Pipe Monitor (PM) and Arbiter Pairs

The three Rx arbiters mentioned above select a head-of-line item across the 32 CGDs only if there is enough room for storing the selected item into the next buffering stage. If some CGDs are full, these CGDs are temporarily removed from the arbiter list of candidate items until room is freed again. CGDs that are temporarily removed from an arbitration point are handled as if there was no pending workload for them (in other words, like an empty list). This mechanism of looking ahead for available room is referred to as *Pipe Monitoring*.

For QoS support, each stage of the pipe consists of a set of an arbiter and a Pipe Monitor (PM), which act together to provide the required QoS performance while preventing head-of-line blocking and reducing to minimum the cross-effect between ports and between CGDs.

In each stage of the pipe, an arbiter is located at the beginning of the stage, selecting which packet is propagated into the next pipe stage. The PM counts the load per CGD and per port of this stage from the arbiter gate until the end of the stage, which can be the next pipe stage arbiter, or NIC's PCIe interface.

The pipe monitors are set of counters that track the load in this pipe stage, per-Port and per-CGDs. When a CGD or a port load reaches its load threshold, The PM signals the units that feed this pipe stage. Upon PM load signal, the related CGD(s) or port(s) are masked out in the arbitration flow of the arbiters of the predecessor units.

Each stage of the pipe listed above is paired with a PM.

8.2.1.2 Rx QoS Units Overview

8.2.1.2.1 Packet's Congestion Domain Mapping in Receive - Overview

The network ports are exposed to the E810 (up to eight ports). Ingress Rx packet processing is done in the E810. The E810 supports two PFC modes: VLAN PFC and DSCP PFC. The PFC modes is a configuration per port.

- **VLAN PFC** — Class of Service (CoS) is based on the 3-bit *PCP* field of the 802.1p header extracted from incoming packet. This value is the packet's UP (User Priority).
- **DSCP PFC** — CoS is based on the DSCP (IPv4)/Traffic Class (IPv6) field. This is a 6-bit value.

8.2.1.2.2 Remapping of the UP Field in Receive

Refer to Section 7.12.6 for the L2 equivalent.

This is a service for the software layer, it acts as a translation service between the software space UP mapping and the Network space UP mapping. The remapping scheme applies also to the traffic destined to EMP (or BMC), which is represented in the tables by its own VSI units.

VLAN UP might be included in the L2 header even when the port is configured for DSCP PFC mode.

8.2.1.2.3 Rx Ports and ETS Arbiters - Overview

The RPB egress arbiter and the RCB arbiter are in charge of Rx QoS support. They manage the Rx processing order through most of the Rx-Pipe. In principal the arbiters implement a three-layer arbitration scheme, one layer Ports arbiter and two layers ETS arbiter:

- **Port arbitration** — This is WRR arbitration between Rx ports. It ensures isolation between the Rx LAN ports. The total available PCIe bandwidth allocated for traffic reception is firstly allocated among the LAN ports. The weight, by default, is proportional to the ports' speed ratio.
- **ETS arbiter** — A two-layer arbitration among CGDs inside each port. The E810 supports two types of CGDs: "Regular" and "High-Priority". In this document, the Regular and High-Priority CGDs are also called "ETS" and "non-ETS", respectively. The regular (or ETS) CGDs involve token-based WRR arbitration. The high-priority (or non-ETS) CGDs get higher priority in each ETS arbiter with no bandwidth limit. The two layers of the ETS arbiters are:
  - **CGD arbitration** — Packet based RR arbitration among all non-ETS CGDs of the port in parallel to bytes based WRR arbitration among all regular CGDs of the port.
    - Each active CGD is associated with one and only one priority. If the CGD is declared as high-priority CGD, it is associated with the high-priority arbiter. Otherwise, it is associated with the regular arbiter.
  - **Strict Priority** — Between the high-priority traffic arbitration winner and the regular traffic arbitration winner. The winner of the arbitration among high-priority CGDs always wins. When the high-priority queue is empty, the regular queues passes.

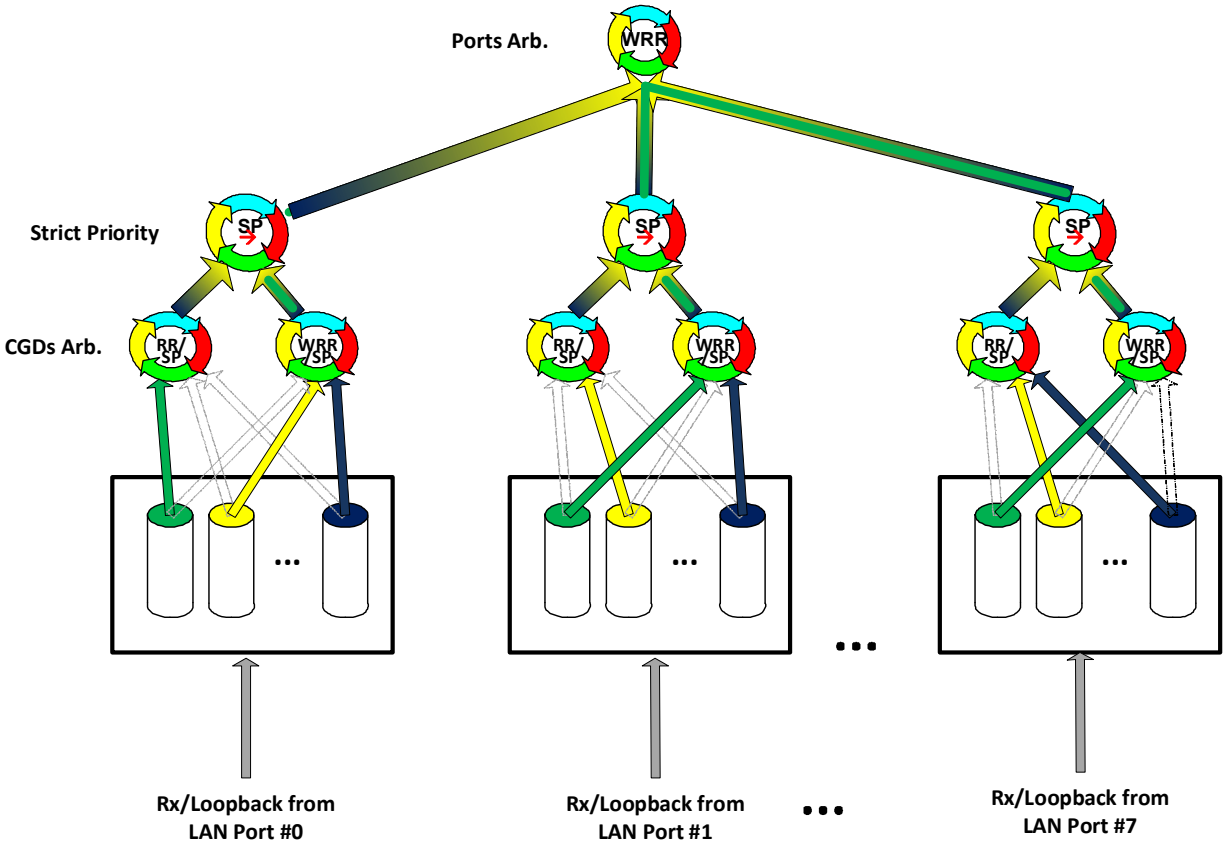


Figure 8-1. Rx QoS Arbiters Unit Scheme

Figure 8-1 shows the structure of CGD arbitration. Inside each port, the high-priority CGDs are associated with the left arbiter, while the low priority CGDs are associated with the right one. The middle arbiter is a simple strict priority arbiter between the winners of WRR arbiters. The upper arbiter is the ports' arbiter.

8.2.1.3 Rx QoS Blocks, Implementation Details and Programming

8.2.1.3.1 Packet's Congestion Domain Mapping in Receive - Implementation Details and Programming

The network ports are exposed to the E810 (up to eight ports). Ingress Rx packet processing is done in the E810.

The E810 supports two PFC modes: "VLAN PFC" and "DSCP PFC".

In both PFC and DSCP modes, congestions are notified using L2 Priority Flow Control (PFC) messages as detailed in Section 3.2.1.5.

8.2.1.3.1.1 Packet CoS Classification Flow in "VLAN PFC" Mode

- Class of Service (CoS) is based on the 3-bit *PCP* field of 802.1p header extracted from the incoming packet. This value is the packet's UP (User Priority).
- When the port is configured to expect multi-TAG packets, the *HAS_UP* bit in GL_SWT_L2TAGCTRL register indicates per-tag type if its UP is a candidate for DCB usage. The UP for DCB is taken from the first tag encountered in the packet that has this bit set. This bit should be set for STags and VLAN (internal and external) EtherTypes.
- The PRTDCB_RUP2TC register controls the mapping of incoming packets to TCs according to the UP field they carry (UP to TC mapping table).
- UP-to-TC mapping (UP2TC) is a table per port, the UP2TC table is configured by the LLDP owner, which can be firmware or software (software might decide to take LLDP ownership, as detailed in Section 8.2.4.5.1). The LLDP owner owns the DCBX as well. UP2TC is configured based on the port's DCB capabilities and DCB setting. The DCBX setting is detailed in Section 8.2.4.5.
- The same mapping is used for packets received from the wires and for those looped back internally. It defines on the account of which TC a packet is stored in the Rx packet buffer, and which UP's bits are set in the PFC XOFF/XON frames issued to the link partner when the filling state of a TC requires it.
- Packets received with no 802.1p tag are also mapped to a TC according the setting made in *NOVLANUP* field of PRTDCB_RUP register that is applied as an input to the UP to TC mapping table of PRTDCB_RUP2TC register. To ensure that LLDP and other MAC Control packets are not dropped internally by the device before they reach EMP or the host, it is recommended that the PRTDCB_RUP.*NOVLANUP* field be set to the No-Drop UP with the lowest index.
- Packet's CGD is calculated in a three-step flow:
  a. The UP of the packet is extracted from the VLAN header of the received packet. Packets received with no 802.1p tag are also mapped to a default UP configured in *NOVLANUP* field of PRTDCB_RUP register.
  b. Packet's TC is extracted based on the UP and per-port UP to TC mapping programmed in the PRTDCB_RUP2TC register.

     As mentioned above, the E810 can be configured as 1-4 port device that can support up to eight TCs per port, or as 5-8 port device that can support a maximum of four TCs per port.

     When the device is configured 5-8 port mode, the PF, when programming the PRTDCB_RUP2TC register, is responsible for configuring all TC values to be limited to 0-3.
  c. Packet CGD == Port num (0-7) * Max TC per port (4 or 8) + TC.

8.2.1.3.1.2 Packet CoS Classification Flow in "DSCP PFC" Mode

- Class of Service (CoS) is based on the *DSCP* (IPv4)/*Traffic Class* (IPv6) field. This is a 6-bit value.
- DSCP is extracted from outer IP header only.
- DSCP configuration is owned by software. DSCP mode is a per-port configuration. Software programs via EMP firmware DSCP translation tables and enables/disables the DSCP mode.
- Software calls EMP firmware using Set Local MIB AQC.
- DSCP-to-UP translation table is a 64-entries LUT that provides a translation from every one of the 64 DSCP values to a 3-bit UP value.

8.2.1.4 Rx-Pipe, Port, and CGD Configuration per Each Usage Model and CGD Type

The NVM is programmed with Rx-Pipe configuration registers and values per each port and DCB topology. The EMP firmware is responsible for reading the correct data set per topology and configuring the pipe accordingly.

8.2.1.4.1 Issuing XOFF and XON

When receiving a packet attached to a No-Drop TC, either from the LAN wire or from the internal loopback path, a PFC XOFF notification can be issued onto the wire and internally to the loopback path for all the UPs mapped to the CGD.

Issuing XOFF behaves differently under different configurations. This section details the different XOFF/XON models per configuration.

- **LFC** — The XOFF/ON mode is a per-port configuration, and the device can be configured to LFC. When the port is configured for LFC mode by the PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE and PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE CSRs, during the link negotiation flow, the device proposes LFC mode for the link partner. If LFC is in agreement with the link partner, the RPB is configured with one CGD for the port with all matching watermark settings. When XOFF/XON needs to be issued, it is done as a LFC message.
- **PFC** — When the port is configured for DCB, a DCBX agent is activated for the port. The DCBX agent runs by default in EMP firmware, Software can take the ownership on LLDP and DCBX. The DCBX is detailed in Section 8.2.4.5. XOFF/XON in this mode are issued as PFC messages.

The format of PFC and LFC messages are detailed in Section 3.2.1.5.

8.2.2 Transmit Path QoS

8.2.2.1 Transmit Path Enhanced Transmission Selection (ETS)

This section discusses the Tx QoS approach in the E810. QoS in transmit includes Tx Scheduling, DCB, and DCBX support, plus the Tx-Pipe balancing to avoid starvations between Tx-Pipe units and between different types of service packets in during the Tx processing pipeline. Special attention is dedicated to burst avoidance as an integral feature required for the COMMS market.

Transmit path ETS is basically managed in the Tx-Scheduler (refer to Section 8.3).

This section deals with the requirements on Tx data path to avoid distortions on the ETS scheme performed by the Tx-Scheduler. The approach relies on two principles:

- Maintain the order of requests issued by the Tx-Scheduler, as much as possible.
- Avoid misleading the scheduler decisions by inexact reports.

Other ETS requirements on Tx data path:

- Serve high-priority traffic as fast as possible in any condition, as long as it is not overtaking its allocated bandwidth.
- Limit packets burstiness beyond what is decided by the Tx-Scheduler.
- Guarantee that best effort TCs can be served at full-blown, regardless of the XOFF/XON events history over lossless TCs.

- Avoid starvation of one ETS traffic class by another, even in the case that it is dropping packets while the other CoSs offer sustained workload.

The section includes:

- Tx-Pipe overview of QoS aspect on the Tx-Pipe.
  - This discussion describes the potential congestion domains, Arbitration and Priority support.
- Transmit QoS, Algorithms, and Configuration.
  - Tx QoS units overview.
  - Tx QoS blocks, implementation details, and programming.
- XOFF/XON methods differences in device operation modes.

Note: Tx-Scheduler is a major partner in the Tx QoS. A dedicated section (Section 8.3, "Transmit Scheduling") details the Tx-Scheduler approach, algorithms, usage models, and configuration.

8.2.3 Tx-Pipe Overview

This section provides an overview for the Tx-Pipe from its QoS aspect. It is not intended to cover the packet processing of the Tx and/or other aspects except the QoS.

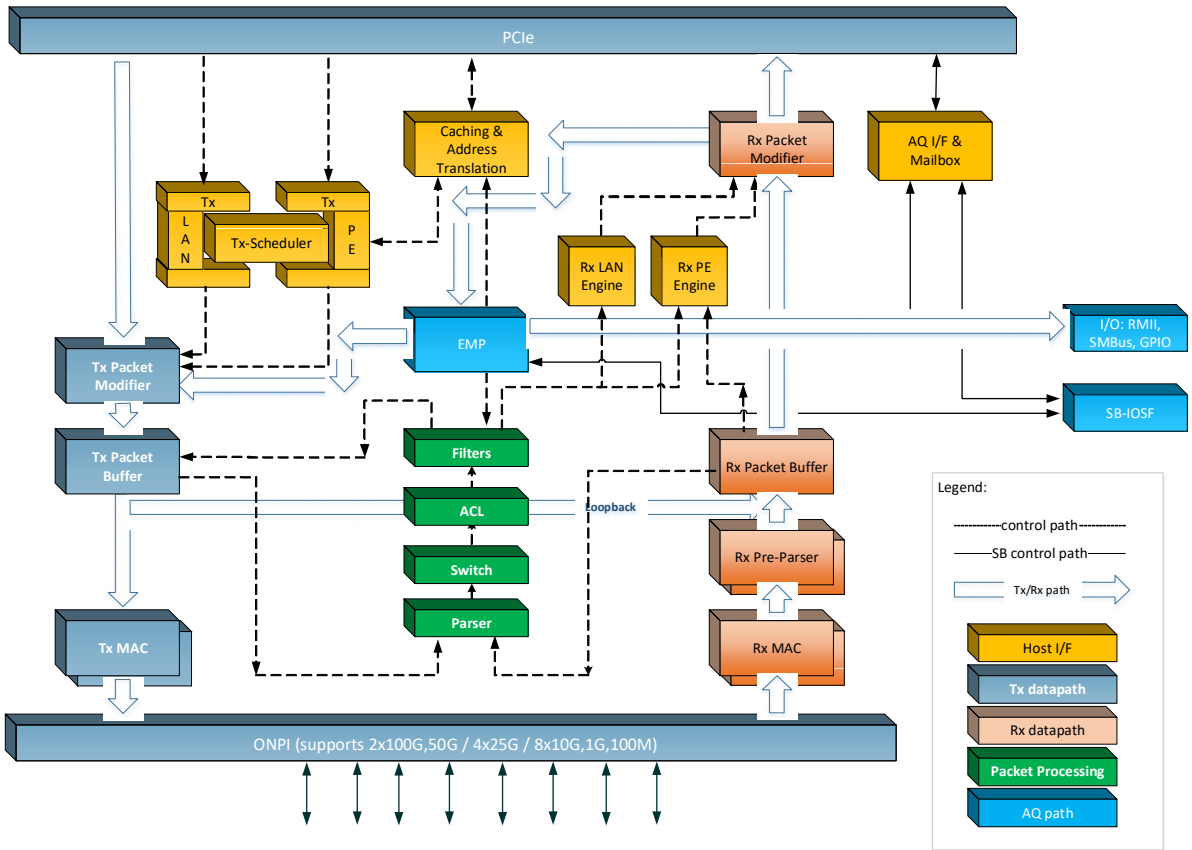


Figure 8-2. E810 Block Diagram

8.2.3.1 Tx Flow - Starting from Scheduling

In the E810, the Tx-Scheduler is located at the top of the Tx-Pipe. The scheduling process is performed prior to fetching the data and descriptors. This enables major memory and power savings as the Tx data and its descriptors are fetched to the device internal memory on demand. Scheduling decisions are made based on the packet sizes digest received by the Tx-Scheduler from the Doorbell messages, and from "Quanta Descriptors", which are explained in detail later in this document. The Advanced Doorbell with packet sizes is described in Section 10.5.6.2.

For a queue that is configured to act in legacy mode, the quanta is pre-configured per queue and is constant. The legacy Doorbell is described in Section 10.5.4.6, and an explanation of "Quanta" is provided in Section 8.3.2.2.1 and in Section 8.3.2.2.2.

The Tx-Scheduler controls and arbitrates both the LAN queue and RDMA transmission. The LAN and RDMA engines wrap the Tx-Scheduler IP. They feed it with transmission requests (Enqueue and Refill flows). When a queue is selected by the Tx-Scheduler, the Tx-Scheduler provides this call to the target unit (Dequeue flow).

8.2.4 Tx-Pipe QoS Next Level of Details

Each LAN Tx-Queue and RDMA QSet is associated with a single TC inside the port space and mapped into a single CGD. To handle sudden PFC XOFF notifications received from the link, the transmit data path is provided with buffers in its different stages. To reduce impact on-die size, provision is made for only two types of traffic: High-Priority (also called Low-Latency (LL) in this Document) and Bulk (B).

Each CGD in Tx is configured with four main behavioral parameters:

- **Advanced Mode Host Interface** — The Advanced Transmit mode enables software to control tightly the burstiness of the traffic from this queue, to interleave transmissions between multiple data sources while limiting the burst size from any one given queue. When a CGS is recognized as Burst Sensitive, all the Tx-Queues associated with this CGD must be configured to operate in Advanced Transmit Mode.
- **Bulk vs. Low Latency** — Each CGD is configured as LL (Low Latency) or B (Bulk) CGD. High-priority CGDs are selected first in the Tx arbiters, allowing them running faster through the Tx-Pipe.
- **Congested vs. Not Congested CGD** — This defines if the specific CGD is declared as Droppable in the network, and therefore no Priority Flow Control (PFC) is expected for the CGD or PFC support for it is required.
- **Congestion Location** — In a regular NIC and in E810 NIC mode, each PFC needs a fast response. To address this, any PFC event results with immediate pausing of the involved CGD traffic anywhere in the Tx-Pipe (that is, stop selecting it in the Tx-Scheduler as well as stop servicing it in TCB and TPB).

Table 8-1 summaries the recommended configuration of each type of traffic in each device mode.

Table 8-1. CGD Configuration for Each Usage

Traffic Type	Service	Host Interface Mode	LL/Bulk	Congested	Congestion Location
Droppable High-Priority	Lower latency relative to regular traffic. No PFC enabled.	Advanced or Legacy Mode	LL	No	Not any
Droppable Regular	No PFC.	Advanced or Legacy Mode	Bulk	No	Not any
No-Drop High-Priority	Lower latency relative to regular traffic. PFC enabled.	Advanced or Legacy Mode	LL	Yes	Tx-Scheduler and Tx-Pipe
No-Drop Regular	PFC enabled.	Advanced or Legacy Mode	Bulk	Yes	Tx-Scheduler and Tx-Pipe

Configuring the CGD for each usage is done by EMP firmware. Software uses the flow detailed in Section 8.2.4.6 to configure each CGD.

8.2.4.1 CoS Translation and Enforcement

The E810 supports two PFC modes: VLAN PFC and DSCP PFC. The PFC modes is a configuration per port.

- **VLAN PFC** — Class of Service (CoS) is based on the 3-bit PCP field of 802.1p header extracts from the incoming packet. This value is the packet's User Priority (UP).
- **DSCP PFC** — CoS is based on the DSCP (IPv4)/Traffic Class (IPv6) field. This is a 6-bit value.

The PFC mode is a per-port configuration. PFC mode can be changed on the fly per software call (port's Tx-Pipe is drained before changing the PFC mode).

Some translate services and enforcement rules are operated in the E810's Tx-Pipe, as detailed in the following paragraphs. For each one of the supported features, its relevancy for PFC mode is marked within each of those paragraphs.

8.2.4.1.1 Remapping of the UP Field in Transmit

Refer to Section 7.12.6 for the L2 equivalent.

This is a service for the software layer. It acts as a translation service between the software space UP mapping and the network space UP mapping. The remapping scheme also applies to the traffic destined to EMP (or BMC), which is represented in the tables by its own four VSI units.

VLAN UP can be included in the L2 header even when the port is configured for DSCP PFC mode.

Note: UP remapping is disabled when the port is configured to DSCP mode. Disabling of UP remapping is done by configuring it to one-to-one mapping, keeping the output to be equal to the input.

8.2.4.1.2 Mapping of UP to TC in Transmit

Register PRTDCB_TUP2TC controls the mapping of UPs to TCs in the transmit path with respect to PFC XOFF/XON notifications received from the link partner or internally from the loopback path. It defines which transmit CGD is paused/released when a UP bit is set in a PFC XOFF/XON frame received (or in internal notification).

Rx and Tx DCB settings are usually identical.

Tx manageability traffic issued by the BMC is bound to the lowest indexed Drop TC, or to TC0 if all TCs are configured as No-Drop.

This mapping is relevant in both PFC and DSCP modes. In DSCP mode, packet classification is based on L3 DSCP field, but the congestion notification is still done by the LLDP PFC messages.

8.2.4.1.3 UP Tag Enforcement in Transmit

In any PFC mode, the E810 enforces software behavior. The CoS of a packet marks the level of service it gets while forwarded through the network. Untrusted software can feed its packets with higher CoS value to get better network performance. This can affect the overall network performance.

In the E810, each Tx-Queue is associated with a single port/TC pair. In its Tx path, the E810 checks every transmitted packet if the CoS value provided by software in the packet's header is allowed for this port/TC.

When the port PFC mode is configured for VLAN PFC, a UP-to-UP translation table is configured per each port/TC pair. For the legal UP values, the output value equal to the input one. For the illegal values, the output value is configured for a default value for the port/TC. The packet is then modified with the correct UP value.

UP tag enforcement can be configured on the fly.

Note: Disabling of UP enforcement is done by configuring it to one-to-one mapping, keeping the output to be equal to the input.

8.2.4.1.4 DSCP Tag Enforcement in Transmit

DSCP enforcement is similar to the UP value enforcement detailed in Section 8.2.4.1.3. One difference is in the DSCP handling resultant from the fact that DSCP is part of L3 header. It is more complex to hardware to modify L3 header of a packet, so the DSCP enforcement is done by either dropping the packet that carries illegal DSCP tag, or raising a malicious event to the PF. The reaction option is configured per-port and it is done using the CSR GL_DCB_TDSCP2TC_BLOCK_IPV[4/6] register.

DSCP tag enforcement is configured per port per CGD, and independently for IPv4 and IPv6 packets.

1. A DSCP table is selected based on:
   a. Port, TC (Part of the Tx-Queue context, piggybacked with each packet through the Tx-Pipe).
   b. Packet's IP type (extracted from packet's header in the query flow).
2. Packet's outermost DSCP tag is extracted and is used as an index to the table selected above.
3. The value read from the table index is a flag that marks if this DSCP tag is legal for this port/TC.
4. If the flag == 0, then:
   a. The packet is dropped if configured so, based on CSR GL_DCB_TDSCP2TC_BLOCK_DIS.
   b. Malicious event is raised if configured so, based on global configuration in CSR GL_MDCK_TX_TDPU.*DSCP_CHECK_FAIL_ITR_DIS.*

Note: CSRs GL_DCB_TDSCP2TC_BLOCK_DIS and GL_MDCK_TX_TDPU.*DSCP_CHECK_FAIL_ITR_DIS* are device-wise configuration registers. They are uploaded from NVM as a global configuration and are not changed by PF.

Table 8-2 summarize the Translation and Enforcement rules per device mode and traffic type. L2 tag insertion and acceptance topics are explained in Section 7.12.6 for the L2 equivalent.

Table 8-2. VM CoS Translation and Enforcement - Summary Table

Traffic Type	UP Remapping	UP Enforcement	DSCP Enforcement	L2 Tags Acceptance	L2 Tags Insertion
Legacy	Yes	Yes	No	Yes	Yes
DSCP	No	No	Yes	Yes	Yes

8.2.4.2 Transmit Flow for ETS

The following flow documents the Tx data path only from an ETS perspective. The ETS Tx-Pipe is described above in Section 8.2.3.

8.2.4.2.1 Pipe Monitor (PM) and Arbiter Pairs

For QoS support, each stage of the pipe consists of a set of an arbiter and Pipe Monitor (PM), which act together to provide the required QoS performance while preventing head-of-line blocking and minimizing the cross-effect between ports and between CGDs.

In Each stage of the pipe, an arbiter is located at the beginning of the stage, selecting which packet is propagated into the next pipe stage. The PM counts the load per CGD and per port of this stage from the arbiter gate until the end of the stage, which can be the next pipe stage arbiter or the NIC's interface.

8.2.4.3 Tx Port and CGD Configuration per Each Usage Model and CGD Type

8.2.4.3.1 CGD Mode of Operation Configuration Parameters

Table 8-3 summaries the recommended configuration of each type of traffic.

Table 8-3. CGD Configuration for Each Usage

Traffic Type	Feature/Control Register Name	NIC Mode			
		Droppable High-Priority	Droppable Regular	No-Drop High-Priority	No-Drop Regular
LL/Bulk	GLDCB_TCB_TCLL_CFG.*LLTC*[0..31] GLDCB_TPB_TCLL_CFG.*LLTC*[0..31]	LL	Bulk	LL	Bulk
Congested	GLDCB_TC2PFC.*TC2PFC*[0..31][1] PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE[Port, UP][2]	No	No	Yes	Yes
Congestion Location	GLDCB_TLPM_IMM_TCB.*IMM_EN*[0..31] GLDCB_TLPM_IMM_TPB.*IMM_EN*[0..31]	Not any	Not any	Tx-Scheduler and Tx-Pipe	Tx-Scheduler and Tx-Pipe
Legacy Host Interface Allowed	GLDCB_TCUPM_LEGACY_TC.*LEGTC*[30..31] GLDCB_TLPM_LEGACY_TC.*LEGTC*[0..31]	Yes	Yes	Yes	Yes

1. This is a multicast register and affects the Tx-Pipe and Rx-Pipe.
2. The indexing of MAC PFC enable is done based on port/UP and not based on CGD.

8.2.4.3.2 Tx-Pipe Arbiters and PM Configuration

8.2.4.3.2.1 Pipe Monitors and Arbiters Configuration per Use Case

Table 8-4. Tx-Pipe Arbiters Configuration in All Ports Setups

	Arbiter Functionality	Control Register Name	Negotiated Speed (Gb/s)			
			100	50	25	10
UPPER PIPE	**Inter Ports Arbitration @ Two Ports Setup**[1]	PRTDCB_TCB_DWRR_QUANTA[0..7].*QUANTA* PRTDCB_TCB_DWRR_SAT[0..7].*SATURATION*	0x14 0x1400	0x0A 0x1400	0x05 0x1400	0x02 0x2000
	Inter Ports Arbitration @ Four or More Ports Setup[2]	PRTDCB_TCB_DWRR_QUANTA[0..7].*QUANTA* PRTDCB_TCB_DWRR_SAT 0..7].*SATURATION*	0x14 0x400	0x0A 0x400	0x05 0x400	0x02 0x400
	Arbitration Between Wait Buffer Release and Regular Traffic in BULK CGDs	GLTCB_BULK_DWRR_REG_QUANTA.*QUANTA* GLTCB_BULK_DWRR_REG_SAT.*SATURATION*	0x1 0x2000			
		GLTCB_BULK_DWRR_WB_QUANTA.*QUANTA* GLTCB_BULK_DWRR_WB_SAT.*SATURATION*	0x2 0x2000			
	Arbitration Between Wait Buffer Release and Regular Traffic in LL CGDs	GLTCB_LL_DWRR_REG_QUANTA.*QUANTA* GLTCB_LL_DWRR_REG_SAT.*SATURATION*	0x1 0x2000			
		GLTCB_LL_DWRR_WB_QUANTA.*QUANTA* GLTCB_LL_DWRR_WB_SAT.*SATURATION*	0x2 0x2000			
LOWER PIPE	**Inter Ports Arbitration @ Two Ports Setup**[1]	TPB_PRTDCB_TCB_DWRR_QUANTA[0..7].*QUANTA* TPB_PRTDCB_TCB_DWRR_SAT[0..7].*SATURATION*	0x14 0x1400	0x0A 0x1400	0x05 0x1400	0x02 0x2000
	Inter Ports Arbitration @ Four or More Ports Setup[2]	TPB_PRTDCB_TCB_DWRR_QUANTA[0..7].*QUANTA* TPB_PRTDCB_TCB_DWRR_SAT[0..7].*SATURATION*	0x14 0x400	0x0A 0x400	0x05 0x400	0x02 0x400
	Arbitration Between Wait Buffer Release and Regular Traffic in BULK CGDs	TPB_BULK_DWRR_REG_QUANTA[0..7].*QUANTA* TPB_BULK_DWRR_REG_SAT[0..7].*SATURATION*	0x1 0x2000			
		TPB_BULK_DWRR_WB_QUANTA[0..7].*QUANTA* TPB_BULK_DWRR_WB_SAT[0..7].*SATURATION*	0x2 0x2000			
	Arbitration Between Wait Buffer Release and Regular Traffic in LL CGDs	TPB_LL_DWRR_REG_QUANTA[0..7].*QUANTA* TPB_LL_DWRR_REG_SAT[0..7].*SATURATION*	0x1 0x2000			
		TPB_LL_DWRR_WB_QUANTA[0..7].*QUANTA* TPB_LL_DWRR_WB_SAT[0..7].*SATURATION*	0x2 0x2000			

1. 1x100G or 2x100G
2. 4x25G, 8x10G, 1x25G + 6x10G, or 2x25G+ 4x10G

Table 8-5. Configuration for Wait Buffer Release Shaping for all Ports Setups and for Upper and Lower Pipe Wait Buffers

	Shaper Control/ Configuration	GLTCB_WB_RL	0x00010020					
PER SPEED SETUP	**@446MHz (High performance SKU - 100G total)**[1]	TCTCB_WB_RL_TC_CFG[0..31] TPB_WB_RL_TC_CFG[0..31]	0x17381	0x17381	0x171C0	0x17167	0x170E0	0x1705A
	@367MHz (Medium SKU - 50G total)[1]	TCTCB_WB_RL_TC_CFG[0..31] TPB_WB_RL_TC_CFG[0..31]	0x17442	0x17442	0x17221	0x171B4	0x17110	0x1706D
	@183MHz (Low Power SKU - 25G total)[1]	TCTCB_WB_RL_TC_CFG[0..31] TPB_WB_RL_TC_CFG[0..31]	0x1788A	0x1788A	0x17445	0x1736A	0x17222	0x170DB

1. Performance level is represented in CSR GL_PWR_MODE_CTL.*CAR_MAX_BW.*

Table 8-6. Tx-Pipe Monitors Configuration in All Ports Setups

	Monitor Functionality	Control Register Name	Negotiated Speed (Gb/s)			
			100	50	25	10[1]
UPPER PIPE	**Per-Port Command Upper Pipe Monitors**	PRTDCB_TCUPM_REG_CTHR[0..7].*PORTOFFTH_L*	0xFFF	0xFFF	0xFFF	0xFFF
		PRTDCB_TCUPM_REG_CTHR[0..7].*PORTOFFTH_H*	0x43C	0x221	0x155	0xBA
		PRTDCB_TCUPM_WAIT_PFC_CTHR[0..7].*PORTOFFTH*	0x9F8	0x5C2	0x42B	0x2F5
	Per-Port Data Upper Pipe Monitors	PRTDCB_TCUPM_REG_DTHR[0..7].*PORTOFFTH_L*	0x98	0x4C	0x26	0x11
		PRTDCB_TCUPM_REG_DTHR[0..7].*PORTOFFTH_H*	0x98	0x4C	0x26	0x11
		PRTDCB_TCUPM_WAIT_PFC_DTHR[0..7].*PORTOFFTH*	0x0	0x0	0x0	0x0
	Per-CGD Command Upper Pipe Monitors	TCDCB_TCUPM_WAIT_CTHR[0..31].*TCOFFTH*	0x4BC	0x2A1	0x1D6	0x13A
	Per-CGD Data Upper Pipe Monitors	TCDCB_TCUPM_WAIT_DTHR[0..31].*TCOFFTH*	0x104	0xB8	0x92	0x7A
LOWER PIPE	**Per-Port Data Lower Pipe Monitors**	PRTDCB_TLPM_REG_DTHR[0..7].*PORTOFFTH_L*	0x1FB	0x109	0xB3	0x65
		PRTDCB_TLPM_REG_DTHR[0..7].*PORTOFFTH_H*	0x1FB	0x109	0xB3	0x65
		PRTDCB_TLPM_WAIT_PFC_DTHR[0..7].*PORTOFFTH*	0x445	0x259	0x1B5	0x110
	Per-CGD Data Lower Pipe Monitors	TCDCB_TLPM_WAIT_DTHR[0..31].*PORTOFFTH*	0x1FB	0x109	0xB3	0x65
PCIe Tx Data		GLDCB_TLPM_PCI_DTHR.*PCI_TDATA*	0x96			

1. The 10 GbE pipe monitor setup values also apply to link speed 5 GbE, 2.5 GbE, 1 GbE and 100 Mb/s.

Note: Some Dedicated RDMA Pipe Monitors must be configured in addition to the above. The RDMA Pipe Monitors setting is detailed in field.

8.2.4.4 Transmit Path Priority Flow Control (PFC)

This section deals with guaranteeing no packet loss at the link partner and inside the E810 further to receiving PFC pause frames from the link partner or further to PFC notifications received from the internal switch.

In each port, TCs are classified in two types relative to PFC: No-Drop TCs (or lossless), and Drop TCs (or best effort delivery).

8.2.4.4.1 PFC Dead-Lock Prevention

Tx traffic that belongs to PFC-enabled TCs can be halted in the Tx-Pipe due to endless XOFF received, either from the line or from the internal loopback path. Flushing out this traffic is a gating condition for the completion of the following operations: Tx-Queue disable, PFR, VFR, disabling the PFC of a TC, or DCBX UP-to-TC remapping.

The device implements a mechanism that automatically flushes out Tx traffic that belongs to a port for which the link goes down. However, there is no similar mechanism for flushing out traffic halted in the Tx data pipe due to endless XOFF conditions. In the case of PFR, an endless XOFF/congestion condition is detected autonomously by the device at the section end and handled as described below. For the other cases (listed above), it is the PF(s) responsibility to detect that the Tx data-path is halted for a long time by periodically monitoring the eight Tx PFC timers attached to a port, one per TC (PRTDCB_TPFCTS.*PFCTIMER*). Each timer (per CGD) is restarted by the device every time the TC is halted by an XOFF notification (received from the link). If any of these counters crosses a threshold defined by the ENDLESS_XOFF_THRESH parameter (in EMP setting module in the NVM), the PF(s) software should take the following actions:

- Post a PFC Ignore admin command (refer to Section 8.2.5.1) for the TC, requesting EMP to set *IGNORE_FC* bit(s) (32 bits - one per CGD) in the GLDCB_TFPFCI register. It causes the entire Tx data path to ignore PFC indications for the concerned TC/port, whether they were received from the line or from the internal loopback path. When in this state, the device flushes out the concerned frames without issuing them over the line or into the internal loopback path.
- When the endless XOFF condition disappeared, and/or when the Tx-Pipe is checked to have been cleaned up (Read PMs for pipe status), the PF clears the *IGNORE_FC* bit by posting a PFC Ignore admin command with *Ignore* flag cleared.

8.2.4.5 Data Center Bridging Exchange Protocol (DCBX)

The DCBX protocol relies on the exchange with the peer of untagged LLDP packets over the physical link.

8.2.4.5.1 DCBX/LLDP Ownership

By default, DCBX is handled by EMP, though the host can decide to take the DCBX ownership once system boot has been completed. Software initiates the transition on a per-port basis by posting the "Stop LLDP Agent" command.

When software owns the DCBX, it can give DCBX ownership back to EMP using "Start LLDP Agent" AQ command. Whenever the host that handles DCBX goes to sleep mode, DCBX is uncovered (EMP does not cover for DCBX). It is the host's responsibility to reset DCB configuration to its default settings (that is, single traffic class, PFC disabled) prior to entering the sleep state. When software handles DCBX, it is also responsible for configuring the DCB settings of the port by setting the DCB registers. In such a case, the end-station can be made by software as the DCBX "master" (the entity that propagates its DCB settings to the peer.

When LLDP/DCB is handled by EMP, the E810 behaves always as a DCBX "slave", retrieving its DCB settings from recommendations or configurations received from the peer.

If LLDP frames are sent by the PF, they are tagged with an outer VLAN tag by the device before issuing them on the lines. In receive, LLDP frames with an outer VLAN tag are directed to the appropriate PF.

Refer to Section 9.8 for the general handling of LLDP packets. The LLDP Agent embedded in EMP (including its DCBX agent) is reset by GLOBR. Further to such reset events, EMP restarts LLDP Agent and DCBX resolution for the ports on which LLDP was handled by EMP.

When LLDP is handled by EMP, following to a CORER event, EMP reconfigures the Tx/Rx paths (and any relevant hardware that was reset) with the DCB configuration that prevailed before the reset occurred.

8.2.4.5.2 DCBX Version

There are two known versions of DCBX standard: IEEE and CEE. The E810 supports both, and can adopt at runtime its peer's mode of operation or to determine the mode of operation based on NVM configuration word: "DCBX Mode".

The NVM configuration selects one of the three options: CEE, IEEE, or adopt peer's mode of operation.

When it configured to adopt peer's mode of operation, it is done based on DCBX TLVs received from the link peer using the flow detailed below.

1. Upon initialization (Linkup, GLOBR, or LLDP ownership hand-off from software to EMP, TLV counters expire) EMP runs IEEE/DCBX as default mode.
2. Upon receiving a DCBX TLVs from the partner:
   — EMP determines the actual DCBX mode it runs. This is done based on the OUI field in the DCBX TLV. In IEEE, the OUI TLV value is "00 80 C2". In CEE, this value is "00 1B 21".
   — If the message contains IEEE/DCBX TLVs, then
      - DCBX mode == IEEE
      - Ignore and CEE/DCBX TLV in the message
   — End If
   — Else If the message contains only CEE/DCBX TLVs, then
      - DCBX mode == CEE
   — End if
3. Whenever the DCBX mode is changed between CEE and IEEE, the whole DCB configuration is reinitialized to the default DCB setting.

Method used by software to find which version EMP runs:

Software sends down "GET_CEE_DCBX_OPER_CFG" AQ command and waits for its completion. Refer to Section 9.8.5.2.2.8 for more details on this command.

If retval == success

DCB Mode = CEE.

Negotiated DCB arbitration available in response buffer for this command.

End if

If retval == ENOENT

DCBX Mode = IEEE

Software sends down "GET_LLDP_MIB" AQ command to find out the DCB arbitration negotiated using DCBX protocol.

End if

If retval == EPERM

Software has taken control of DCBX. Query software DCB agent for any DCBX configuration

End if

8.2.4.5.2.1 CEE vs. IEEE DCBX

This section highlights the main difference between CEE and IEEE modes that affects the DCBX resolution. Refer to the two standard specifications for more details.

Operational Configuration:

IEEE DCBX TLVs always advertise the operational setting of the feature. CEE DCBX, on the other hand, computes the operational configuration based on local DesiredCfg and peer's DesiredCfg. If software wants to determine the OperCfg, it cannot use the "GET_LLDP_MIB" AQ command, for CEE as OperCfg is not the wire in case of CEE. Software must use the "GET_CEE_DCBX_OPER_CFG" AQ command for obtaining the OperCfg in CEE. Refer to Section 9.8.5.2.2.8 for detail on this command.

Priority Groups:

CEE DCBX has a concept of Priority Group, which is not present in IEEE DCBX. EMP is required to equate the Priority Group and Traffic Class concepts. That is, treat the Priority to Priority Group mapping received in the CEE DCBX TLV as the Priority to Traffic Class mapping as well.

Priority Flow Control:

For CEE DCBX, if the PFC does not become operational via CEE DCBX state machine, CEE DCBX disables PFC and enables link-level flow control. When EMP runs the DCBX agent in IEEE mode, this recommendation is irrelevant since it behaves in slave mode and aligns itself to the peer.

8.2.4.5.3 DCBX Managed Objects

When a port DCBX is handled by EMP, the DCBX Managed Objects are the DCBX parameters stored internally in EMP data RAM and/or in device registers. All DCBX objects are per LAN port. Two instances are stored per port: one issued by the device to the link partner referred as Local DCBX parameters, and one received from the link partner referred as Remote DCBX parameters.

When DCBX is handled by EMP, the PFs must restrain themselves from any write access to the DCB registers listed in Table 8-7. "RW" in the table refers to the EMP capability to write the object.

Table 8-7. Local DCBX Managed Objects

IEEE Object Name *CEE Object Name	Data Type	Width (in Bits)	Admin	Default
ETS Configuration TLV				
Willing	Boolean	1	RO	1
Credit-Based Shaper (CBS)	Boolean	1	RO	0
Max TCs *Num TCs Supported	Unsigned Integer	3	RW	0[1]
Priority Assignment Table *Priority Allocation Table	Unsigned Integer [0..7]	8x4	RW	0
TC Bandwidth Table *Priority Group Allocation Table	Unsigned Integer [0..7]	8x7	RW	0
TSA Assignment Table	Unsigned Integer [0..7]	8x8	RW	0[2]
PFC Configuration TLV				
Willing	Boolean	1	RO	1
MBC	Boolean	1	RO[3]	0
PFC Cap *Num TC PFC Supported	Unsigned Integer	4	RO	0x8[4]
PFC Enable *PFC Config Table	Boolean [0..7]	8x1	RW	0
Application Priority Configuration TLV				
Application Priority Table *App Protocol Config Table	Unsigned Integer	32x24	RW	0
MFS per TC Table	Unsigned Integer	8x15[5]	RW	1536

1. 0 is the encoding for 8.
2. Strict Priority algorithm is assumed by default (indicated by a zero value) on each user priority. Setting the value of 2 selects ETS bandwidth allocation scheme.
3. MACsec support.
4. The E810 is always able to support up to eight PFC-enabled traffic classes, though in some cases (when Jumbo is enabled) fully independent PFC behavior is partially achieved.
5. Max Frame Size is defined in bytes.

8.2.4.6 DCB Configuration Using MIB TLVs

The E810 manages internally a wider set of MIB TLVs, which allows configuring Tx and Rx DCB pipes and DSCP settings. Software configures the DCB hardware for both VLAN and DSCP modes using the same method by calling the AQCs "Get Local LLDP MIB" and "Set Local LLDP MIB". In addition, independent and optionally asymmetric setup for Tx and Rx is available using the same method.

Table 8-8 Provides the list of all the TLVs used for DCB configuration.

Table 8-8. DCB Configuration MIB TLVs

		TLV ID	Description	Comments
VLAN PFC MODE	Multicast (Rx and Tx)	0x09	ETS Configuration	Standard DCBX TLVs and OUI. Exchangeable with the link partner. Every "operational TLV" update in this section is immediately copied by EMP firmware to both Rx and Tx corresponding TLVs.
		0x0A	ETS Recommendation	
		0x0B	PFC per UP	
		0x0C	Application Priority	
	Rx	0x89	ETS Configuration	Separate Rx and Tx configuration TLVs. OUI == 0xFF_FF_FF Those TLVs are not exchanged in LLDP. Any update of a TLV in this section is translated to a hardware configuration of Rx or Tx DCB pipe. These TLVs are updated when the operational TLVs are updated. Software can set separately Rx or Tx TLVs (AQC Set Local MIB TLV) to activate different setup for Tx and Rx.
		0x8B	PFC per UP	
		0x8C	Application Priority	
	Tx	0x99	ETS Configuration	
		0x9B	PFC per UP	
		0x9C	Application Priority	
DSCP PFC MODE	Multicast (Rx and Tx)	0x41	DSCP to UP	OUI == 0xFF_FF_FF Those TLVs are not exchanged in LLDP. Every "Multicast TLV" update in this section is immediately copied by EMP firmware to both Rx and Tx corresponding TLVs.
		0x42	DSCP Enforcement	
		0x43	Bandwidth per TC and TC priority	
		0x44	PFC per UP/TC (Same ID)	
	Rx	0x45	DSCP to UP	Separate Rx and Tx configuration TLVs. OUI == 0xFF_FF_FF Those TLVs are not exchanged in LLDP. Any update of a TLV in this section is translated to a hardware configuration of Rx or Tx DCB pipe. These TLVs are updated when the operational TLVs are updated. Software can set separately Rx or Tx TLVs (AQC Set Local MIB TLV) to activate different setup for Tx and Rx.
		0x46	Bandwidth per TC and TC priority	
		0x47	PFC per UP/TC (Same ID)	
	Tx	0x48	DSCP Enforcement	
		0x49	Bandwidth per TC and TC priority	
		0x4A	PFC per UP/TC (Same ID)	

The format of the Rx (0x89, 0x8B, or 0x8C) and Tx (0x99, 0x9B, or 0x9C) TLVs are identical to the corresponding standard DCBX TLVs.

The rest are of non-standard TLV formats are as detailed below.

Notes and Restrictions:

- When software configures the Rx or the Tx separately (using the Rx or the Tx MIBs), Tx-Scheduler is configured with the Tx-Pipe according to the Tx MIBs. Software is responsible for issuing "Stop LLDP Agent" prior to configuring Rx and Tx separately.
- When a port acts in DSCP mode, DCBX protocol is disabled. Software is responsible for issuing "Stop LLDP Agent" prior to configuring a port for DSCP mode.
- When software feeds LLDP MIB, VLAN PFC mode TLVs "ETS configuration" and "PFC per UP" are configured as pairs. If software tries to program one without the other in a single AQ call, this call returns with the error code EINVAL.
- Same restriction and behavior for the DSCP. TLVs "Bandwidth per TC and TC priority" and "PFC per UP/TC" are pairs.
- If software tries to set the DSCP-related MIB while PFC mode != DSCP, the command is rejected with the error code EMODE. Likewise, if software tries to set the VLAN-related MIB while PFC mode != VLAN, the command is rejected with the error code EMODE.

8.2.4.6.1 UP-to-TC Mapping Configuration

The configuration of UP-to-TC for Rx is done using CSR PRTDCB_RUP2TC. EMP firmware uses this CSR when the Rx UP2TC is changed as part of the DCBX handling, or as a response to a software call for Set Local MIB (Rx only or multicast).

The configuration of UP-to-TC for Tx is done using CSR PRTDCB_TUP2TC. EMP firmware uses this CSR when the Rx UP2TC is changed as part of the DCBX handling or as a response to a software call for Set Local MIB (Tx only or Multicast).

8.2.4.6.2 DSCP-to-UP - Subtype 0x41 or 0x45

Table 8-9. DSCP-to-UP Format

TLV Type = 127 (7 bits)	TLV Information String Length = 148 (9 bits)	OUI FF-FF-FF (24 bits)	Subtype = 0x41 or 0x45 (8 bits)	DSCP-to-UP Table Table 8-10 (145 bytes)

Table 8-10. DSCP-to-UP Configuration

Name	Byte.Bit	Value	Remarks
IPv4 DSCP-to-UP mapping used for Rx. Affects packets coming from the network or form loopback (VM to VM).			
DSCP #0	0	UP value	Configured UP value for DSCP #0
DSCP #1	1	UP value	Configured UP value for DSCP #1
: :	: :	: :	
DSCP #63	63	UP value	Configured UP value for DSCP #63
Untagged	64	UP value	Configured UP value for Untagged (non-IP packets)
IPv6 DSCP-to-UP mapping used for Rx. Affects packets coming from the network or form loopback (VM to VM).			
Reserved	65-79		Padding for alignment.
DSCP #0	80	UP value	Configured UP value for DSCP #0
DSCP #1	81	UP value	Configured UP value for DSCP #1
: :	: :	: :	
DSCP #63	143	UP value	Configured UP value for DSCP #63

8.2.4.6.3 DSCP Enforcement - Subtype 0x42 or 0x48

Table 8-11. DSCP Enforcement Format

TLV Type = 127 (7 bits)	TLV Information String Length = 132 (9 bits)	OUI FF-FF-FF (24 bits)	Subtype = 0x42 or 0x48 (8 bits)	DSCP Enforcement Table Table 8-12 (128 bytes)

Table 8-12. DSCP Enforcement Configuration

Name	Byte.Bit	Value	Remarks
IPv4 DSCP enforcement[1]			
Port's TC#0 DSCP	0-7	Bitmap	Bit per DSCP value, marks if Block Tx of Packet uses this DSCP.
Port's TC#1 DSCP	8-15	Bitmap	Bit per DSCP value, marks if Block Tx of Packet uses this DSCP.
Port's TC#2 DSCP	16-23	Bitmap	Bit per DSCP value, marks if Block Tx of Packet uses this DSCP.
Port's TC#3 DSCP	24-31	Bitmap	Bit per DSCP value, marks if Block Tx of Packet uses this DSCP.
Port's TC#4 DSCP	32-39	Bitmap	Bit per DSCP value, marks if Block Tx of Packet uses this DSCP.
Port's TC#5 DSCP	40-47	Bitmap	Bit per DSCP value, marks if Block Tx of Packet uses this DSCP.
Port's TC#6 DSCP	48-55	Bitmap	Bit per DSCP value, marks if Block Tx of Packet uses this DSCP.
Port's TC#7 DSCP	56-63	Bitmap	Bit per DSCP value, marks if Block Tx of Packet uses this DSCP.
IPv6 DSCP enforcement[1]			
Port's TC#0 DSCP	64-71	Bitmap	Bit per DSCP value, marks if Block Tx of Packet uses this DSCP.
Port's TC#1 DSCP	72-79	Bitmap	Bit per DSCP value, marks if Block Tx of Packet uses this DSCP.
Port's TC#2 DSCP	80-87	Bitmap	Bit per DSCP value, marks if Block Tx of Packet uses this DSCP.
Port's TC#3 DSCP	88-95	Bitmap	Bit per DSCP value, marks if Block Tx of Packet uses this DSCP.
Port's TC#4 DSCP	96-103	Bitmap	Bit per DSCP value, marks if Block Tx of Packet uses this DSCP.
Port's TC#5 DSCP	104-111	Bitmap	Bit per DSCP value, marks if Block Tx of Packet uses this DSCP.
Port's TC#6 DSCP	112-119	Bitmap	Bit per DSCP value, marks if Block Tx of Packet uses this DSCP.
Port's TC#7 DSCP	120-127	Bitmap	Bit per DSCP value, marks if Block Tx of Packet uses this DSCP.

1. DSCP values enforcement. A per-CGD 64-bit bitmap. Signs per CGD per DSCP value if packet carries DSCP value X when the queue associated with CGD Y needs to be Blocked. Affects packets going to the network or to loopback (VM to VM). The behavior of the device in case of wrong DSCP value is configured by CSR GL_DCB_TDSCP2TC_BLOCK_DIS and GL_MDCK_TX_TDPU.DSCP_CHECK_FAIL_ITR_DIS as explained in Section 8.2.4.1.4.

8.2.4.6.4 DSCP Bandwidth per TC and TC Priority 0x43, 0x46, or 0x49

Table 8-13 details the DSCP bandwidth per TC and TC priority MIB TLV. Its format is identical to IEEE DCBX ETS configuration TLV. The irrelevant fields are marked as reserved.

Table 8-13. DSCP Bandwidth per TC and TC Priority Configuration Format

TLV Type = 127 (7 bits)	TLV Information String Length = 25 (9 bits)	OUI FF-FF-FF (24 bits)	Subtype = 0x43, 0x46, or 0x49 (8 bits)	RSV (1 bit)	CBS (1 bit)	RSV (3 bits)	Max TCs (3 bits)	Reserved (4 bytes)	TC B.W. Table (8 bytes)	TSA Assignment Table (8 bytes)

8.2.4.6.5 PFC per UP/TC 0x44, 0x47, or 0x4A

Table 8-14 details the PFC enable TLV. Its format is identical to IEEE DCBX Priority-based Flow Control TLV. The irrelevant fields are marked as reserved.

Table 8-14. PFC per UP/TC Configuration Format

TLV Type = 127 (7 bits)	TLV Information String Length = 6 (9 bits)	OUI FF-FF-FF (24 bits)	Subtype = 0x44, 0x47, or 0x4A (8 bits)	Reserved (1 byte)	PFC Enable (1 byte)

8.2.5 DCB Admin Commands

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

8.2.5.1 PFC Ignore (0x0301)

This command is used to request the device to ignore PFC condition present on a Tx path for a TC. The same command is used to release PFC ignore request. When PFC packets are ignored, they are considered as regular packets and can be forwarded to host.

Table 8-15. PFC Ignore Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0301	Command opcode.
Datalen	4-5		Must be zeroed by driver.
Return Value/VFID	6-7		Must be zeroed by 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	16-17		**Bits 16.7-16.0:** TC Bitmap Bitmap of the TCs concerned by the command. Bit n set to 1b means TC n is concerned by the request. When LFC is used instead of PFC, TC index 0 is used to request ignoring LFC. **Bits 17.6-17.0:** Reserved, must be zeroed. **Bit 17.7:** Ignore Flag 0b = The PF requests to release any ignore PFC condition request issued for the TC indexes set to 1b in the TC Bitmap. 1b = The PF requests to ignore PFC conditions on the TC indexes set to 1b in the TC Bitmap.
Reserved	18-31		Reserved. Must be zeroed.

Table 8-16. PFC Ignore Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0301	Command opcode.
Datalen	4-5		Must be zeroed.
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.
Completion_flags	16		**Bits 16.7-16.0:** TC Status Bitmap Returns the TCs for which PFC is currently ignored. Bit n set to 1b means PFC condition on Tx path for TC n is ignored by the device.
Reserved	17-31		Reserved. Must be zeroed.

8.2.5.2 Query PFC Mode (0x0302)

This AQ call returns an indication if DSCP-based PFC or VLAN-based PFC is enabled.

This is a Direct Admin Queue command with additional command attributes, and completion attributes are provided within the data buffer. Table 8-17 describes command format and defines command-specific fields.

Table 8-17. Query PFC Mode Command and Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1 for details.
Opcode	2-3	0x0302	Command opcode.
Reserved	4-5		Reserved.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
PFC Mode	16		Zeroed by the Driver. Written by firmware provides the PFC mode setup. 0 = DCB is disabled for this port (by an NVM setting). 1 = VLAN based PFC is enabled. 2 = DSCP based PFC is enabled.
Reserved	17-23	0	Reserved. Must be set to 0.
Data Address High	24-27		0
Data Address Low	28-31		

8.2.5.3 Set PFC Mode (0x0303)

This AQ call configures the PFC mode to DSCP-based PFC mode or VLAN-based PFC.

This is a Direct Admin Queue command with additional command attributes, and completion attributes are provided within the data buffer. Table 8-18 describes command format and defines command-specific fields.

If DCB is disabled for this port by NVM setting, EMP firmware ignores this command, and in the response buffer the PFC mode is set to 0 “DCB is disabled for this port (by an NVM setting)”.

As part of flow of this command, EMP firmware resets all PFC parameters of both VLAN and DSCP modes. After setting the PFC mode, LLDP exchange or software AQ call can configure the PFC parameters.

Table 8-18. Set PFC Mode Command and Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1 for details.
Opcode	2-3	0x0303	Command opcode.
Reserved	4-5	0	Reserved.
Return Value/VFID	6-7		Return value, Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
PFC Mode	16		In Command: 0 = Reserved. 1 = VLAN based PFC is enabled. 2 = DSCP based PFC is enabled. In Response: 0 = PFC is not configured for this port. 1 = VLAN based PFC is enabled. 2 = DSCP based PFC is enabled.
Reserved	17-23	0	Reserved. Must be set to 0.
Data Address High	24-27		0
Data Address Low	28-31		

8.2.5.4 Set DCB Parameters (0x0306)

This command is used to request that the firmware apply the DCB configuration even when LLDP/DCBX is disabled.

Table 8-19. Set DCB Parameters Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1 for details.
Opcode	2-3	0x0306	Command opcode.
Reserved	4-5	0	Must be zeroed by driver.
Return Value	6-7		Must be zeroed by 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:** Link-up DCB configuration mode for the port 0b = Firmware does not apply any DCB configuration when a link up event occurs. 1b = Firmware applies the default DCB configuration when a link up event occurs. **Bit 1:** Persistently set the DCB configuration mode for the current port: 0b = No persistence of the DCB configuration mode settings 1b = Persistence for the selected DCB configuration mode.
Valid Flags	17	0	**Bit 0:** Bit 0 of the command flags is valid If the Valid Flag bit is 0, no change is made to the Link-up DCB configuration mode. **Bit 1:** (Valid only when Bit 0 is set to 1b.) 0b = Bit 1 of the command flags is not valid. 1b = Bit 1 of the command flags is valid.
Reserved	18-31	0x0	Reserved.

Table 8-20. Set DCB Parameters Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0306	Command opcode.
Datalen	4-5	0	No external buffer.
Return Value	6-7		Return code written by firmware. In the case of a failure, the existing configuration is unmodified.
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 8-21. Set DCB Parameters Expected Behavior

	Command Flags = 0	Command Flags = 1
DCBX Enabled	No change to existing behavior.	No change to existing behavior.
DCBX Disabled	No change to existing behavior (i.e., don't touch).	If software has not used Set Local LLDP MIB to change the MIB since GLOBR, apply the default settings. If software has changed the MIB, do nothing since software will handle it.

8.2.6 LLDP/DCBX Admin Commands

Refer to the LLDP Protocol commands described in Section 9.8.5.2.2.

8.2.6.1 LAN Queue Overflow Event (0x1001)

This event is sent from the device to a PF. It does not generate a response.

Table 8-22. LAN Queue Overflow Event

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x1001	Command opcode.
Datalen	4-5	0x0	N/A
Return Value	6-7	0x0	N/A
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.
PRTDCB_RUPTQ	16-19		Contains a copy of the PRTDCB_RUPTQ register reporting the absolute index (in the device space) of the reported receive queue.
QTX_CTL	20-23		Contains a copy of the QTX_CTL register of the matched transmit Queue Pair of the reported receive queue.
Reserved	24-31		Reserved. Must be zeroed.

Note: Software can reset the problematic function or resolve the issue in any other way. When the issue is resolved or the function is reset, the PF must clear the relevant timer by writing a 1b to the corresponding bit in CSR GLDCB_RCGDTI.

8.3 Transmit Scheduling

8.3.1 Hierarchical Scheduling

The E810 supports up to 16K Tx-Queues. The software manages packet enqueues for transmission to any one of the Tx-Queues, at any time, and in any order. For information on the Tx-Queue, refer to Section 10.5.5. Each Tx-Queue is associated with a single software entity (PF or VF), CPU core, and usage type called a user priority. The Tx-Scheduler is responsible for controlling the transmission of all Tx-Queues. It executes a scheduling algorithm and decides which the Tx-Queue is allowed to transmit and how many bytes. To meet all QoS agreements while considering the network capacity, and to reduce packet drop or flow control events, the Tx-Scheduler takes into account the following configuration parameters:

- Priority.
- Bandwidth allocation, bandwidth limit, minimum guaranteed bandwidth.
- Network topology, the network infrastructure fed by device ports, its capacity, and the deployment of the Rx units (switches, routers, target NICs, and so on) in the network.

- Those Rx units might be buffer-limited or limited by packet processing power. To address both types, the E810 can configure each topology node to arbitrate and shape for bytes per second (BPS) or packets per second (PPS).
- Quality of Service (QoS) and service level agreement (SLA).

The E810 Tx-Scheduler configured topology reflects the network topology to which the port is connected from one side, and the software functions (PFs, VFs, VMs, users, and so on) running on the Host on the other side. The network topology is ordered in layers, each layer representing a physical junction of the network. Each junction is also declared with its capacity that is derived by the line speed, the buffer sizes of the relevant routers, and so on.

Multiple Tx-Queues can feed every line in the network. These queues are also ordered in groups similar to VMs. The VMs themselves can be ordered in groups similar to PF.

The network resembles a tree, where each layer represents a physical junction (a router or a logical entity) with its QoS requirement. Section 8.3.3 shows some tree topology examples that can be used to represent the network topology in use cases such as cloud, enterprise, and COMMS.

To satisfy the QoS requirements of each layer of grouping and the deployment of the network, the Tx-Scheduler performs tree-based scheduling, which takes into account the QoS requirements and deployment limits of the network.

Tx-Scheduler Responsibilities:

The Tx-Scheduler accounts for the bandwidth allocation, bandwidth limit, and minimum bandwidth settings for payload and all headers generated by software and hardware. This is true for LAN traffic and payload as well as for headers generated by the Protocol Engine for RDMA traffic. It also includes the L2 tags, Ethernet, MPA, MPA Markers, and RDMA inserted by hardware on-the-fly. If hardware TEP is supported in Tx, the Tx-Scheduler also accounts these additional headers.

Section 8.3.2 describes the scheduling concept and its terms.

To support multiple usage models, abstract hardware configuration tables are used for the Tx-Scheduler and the usage model requirements are achieved by software and EMP firmware configuration parameters.

8.3.2 Hierarchical Scheduling Concept

8.3.2.1 Abstract Scheduling Tree

This section describes tree scheduling, its concept, terminology, and logical scheduling tree structure. The tree consists of the following components (examples relate to Figure 8-7):

- **Tree Node (or Node)** — The basic tree element. Each Tree Node can have at most one Parent Node, and one or more Child Nodes. The tree configuration establishes Parent/Child relations between all the Tree Nodes.
- **Root Node** — A Tree Node lacking a Parent Node. It occupies the lowest level on the tree - Level 0 (Node00 in this example). The Root Node is the end point of every scheduling flow.
- **Parent Node** — A Tree Node with one or more Child Nodes. Parent Nodes are always located at a lower tree level than Child Nodes. Node10 is a Parent Node, with two Child Nodes, Node20 and Node21. A Parent Node leads to its Child Nodes at scheduling tree traversal.
- **Child Node** — A Tree Node that has a Parent Node. A Child Node might be a Parent Node with respect to other Nodes. Node20 is a Child Node to Node10, and Parent Node with respect to Node300.

- **Sibling Nodes** — Tree Nodes that are Child Nodes of the same Parent Node. All Sibling Nodes are located on the same Tree Level. Nodes Node20 and Node21 are Sibling Nodes, whose Parent Node is Node10.
- **Leaf Node** — A Tree Node that does not have any Child Nodes. Leaf Nodes are located on the highest level of the Tree. Node300 is a Leaf Node. Every scheduling flow starts from a Leaf Node.
- **Tree Level** — A tree elevation. All the Tree Nodes located on the same tree elevation are at the same Tree Level. However, nodes on the same level do not necessarily have same Parent Node, since they might belong to different subtrees. Nodes Node21 and Node22 are located on the same Tree Level (Level 2), but they belong to different Subtrees, since they have different Parent Nodes, the parent of Node21 is Node10 and that of Node22 is Node11.
- **Subtree** — A portion of the tree comprised of the Tree Node as a Root Node and its descendants. The Subtree of Node10 includes Node20, Node21, Node300, Node301, Node302 and Node303.
- **Height of the Tree** — Number of Tree Levels + 1. Height of the tree is a number of steps in the scheduling tree traversal.
- **Blocked Node** — A node is considered to be blocked if it cannot satisfy the applied scheduling constraints or if all its Child Nodes are blocked. A Leaf Node is also defined as a blocked node when no work is available in its associated transmit descriptor queue or its RDMA QSet. Blocked Nodes cannot be included in scheduling tree traversal.
- **Branch** — In the context of Scheduling Tree, a branch is the pass from a Node to the Root Node. The Branch of a Node is its Parent's Node ID, Grandparent's Node, and so on, up until the Port's Root Node ID. The Branch of Node310 is Nodes list: Node24, Node11 and Node0. The Branch of Node20 is the list of Node10 and Node0. Most of the Scheduler operations are iterative walk from the Leaf Node to its root. Tx-Scheduler keeps the branch data of every Leaf Node in tables called "Branch Tables".

8.3.2.2 Scheduling Configuration Terms

8.3.2.2.1 Scheduling Quanta

As the network bandwidth exceeds 10 Gb/s, packet-by-packet scheduling requires the Tx-Scheduler to complete scheduling decisions within a few nanoseconds - this is not trivial. The E810's Tx-Scheduler schedules a "quanta" in each scheduling decision. A "quanta" is a configurable amount of bytes that can consist of one or more packets. The quanta size is configured per Tx-Queue and represents the number of transmitted bytes per scheduling cycle. The quanta size per queue varies from 250 bytes to 4 KB. The lower bound depends on LAN speed and topology configuration.

8.3.2.2.2 Configured Quanta vs. Actual Quanta Size

As noted above, a quanta is a number of bytes configured for each queue to be considered a single scheduling unit. The transmission unit must also take into account the boundaries of the transmitted packets - the device must not transmit a partial packet.

Packet boundaries are not synchronized with quanta boundaries. Therefore, to avoid transmitting partial packets, the packet queuing software flow is as follows:

- When a packet enters a Tx-Queue, its actual size is accumulated to the current active quanta of the queue.
- When the accumulated size of the current quanta exceeds the "per queue configured quanta size", the current quanta is locked and next packets are aggregated in the next quanta unit.

- The "Actual Quanta" is the accumulated size of the packets that are included in the quanta. Typically, the actual quanta size is larger than configured quanta size and it fits the packet boundaries.
- The actual quanta might include one or more packets. This quanta is queued to the Tx-Scheduler as a one unit request. The Tx-Scheduler grants this quanta to the queue as one unit.

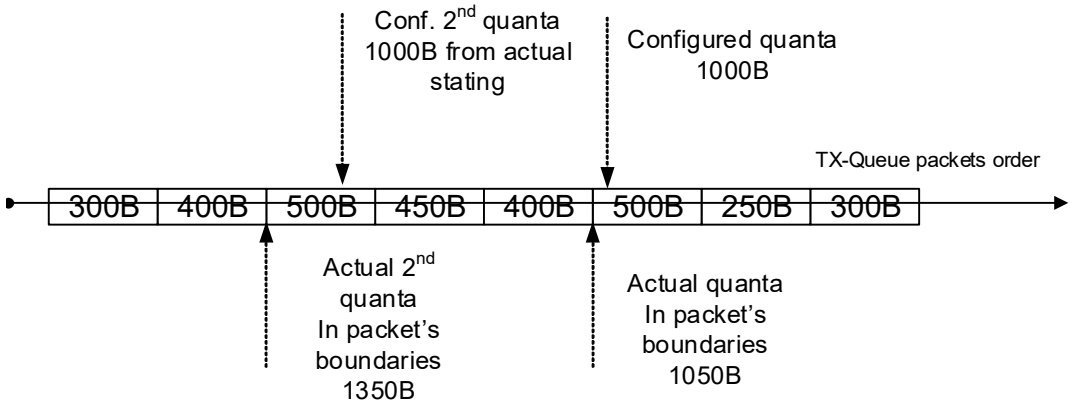


Figure 8-3. Configured Quanta and Actual Quanta Example

8.3.2.2.3 Bandwidth Allocation, Minimum Bandwidth Guarantee, and Bandwidth Limit

The E810 supports various bandwidth distribution mechanisms that can be applied to any node in the scheduling tree. These mechanisms define how Parent Node resources are used and shared by the Child Nodes. Those constrains are stated in the terms of bandwidth.

- **Relative Bandwidth Allocation or Bandwidth Share** — Allows to specifying a relative share of available bandwidth among Sibling Nodes. Sibling Nodes share the total amount of bandwidth available from the Parent Node. The share of the Parent bandwidth is relative, both with respect to the total amount of bandwidth available, and use of the bandwidth by other Sibling Nodes. If one of the Sibling Nodes cannot use its bandwidth share, the unused bandwidth can be utilized by other Sibling Nodes. Bandwidth share can be expressed in terms of weights or credits assigned to each tree Node.
- **Bandwidth Limit (PIR - Peak Information Rate) - Also known as Rate Limit (RL)** — Allows specifying a maximum bandwidth to be consumed by the Tree Node. RL is an absolute number specifying a maximum amount of byte or packets a Node can consume during the course of one second. Rate limit guarantees that a link does not oversaturate the receiver on the remote end, and also enforces an SLA between the subscriber and network provider.

  A Child Node cannot consume more bandwidth than the limit of its Parent Node. However, the sum of the bandwidth limits of the Child Nodes can exceed the bandwidth limit of the Parent Node, potentially enabling better utilization of the available bandwidth. Figure 8-7 illustrates this concept, where Child Nodes are represented by overlapping virtual pipes contained within the Parent Node virtual pipe.

  Each tree node can have an individually-assigned bandwidth limit. Some Nodes in the tree might need to share the Bandwidth Limit and use the Shared Rate Limiter scheme, explained below.

- **Shared Bandwidth Limit** — Allows specifying a bandwidth limit for a group of nodes that are not co-located in the tree, for example Child Nodes of different parent nodes. Another example of shared bandwidth usage is if you want to limit the bandwidth for a subgroup of a sibling group. You can define one sibling group that shares bandwidth allocation and still limit each subgroup individually.

  A node cannot be configured for both an individual bandwidth limit and a shared bandwidth limit at the same time. An exception for that statement is the ability to associate a node with a shared bandwidth limit group, but to configure it for credit contribution only. This means that this node contributes its bandwidth limit unused credits to the group but it is limited to its individual limit configuration. This is detailed later in this section.

- **Maximum Burst Size** — Specifies the maximum number of bytes that can be transmitted in a short period of time. When a node is configured with a bandwidth limit or a group is configured with shared bandwidth limit, the maximum burst size specifies the maximum amount of credits (in bytes) this rate limiter can collect (by the leaky bucket algorithm) when it is not transmitted for any reason (such as no work available for this node or other node with higher priority...).

- **BPS Scheduling vs. PPS Scheduling** — As stated in Section 8.3.1, the Tx-Scheduler configuration must take in account, among other parameters, the receive capabilities of the network nodes. A network node's receive capabilities can be declared as Rx packets buffering limited or as receive pipe processing limited. To address both limits, any node in the E810's Tx-Scheduler, can be configured to "Bytes Per Second" mode or to "Packets Per Second" mode. This configuration is applied to both bandwidth allocation and bandwidth limit.

In terms of configuration, each sibling group is configured either for BPS or for PPS mode. As an implementation, when a scheduling request is fed to the Tx-Scheduler, it includes both the quantity of bytes and number of packets. When this request is propagated through the Tx-Scheduler pipe, any node that is configured to PPS considers the request as bytes request of the number of requested packets multiplied by a constant number. From this point onwards, scheduling and shaping of PPS is similar to BPS shaping.

- **Minimum Bandwidth Guarantee (CIR - Committed Information Rate)** — Allows specifying a minimum bandwidth allocated to a tree node when it is not blocked. Like PIR (detailed above), CIR specifies an absolute bandwidth. While PIR defines the maximum bandwidth the node can consume, the CIR marks committed bandwidth for the Node.

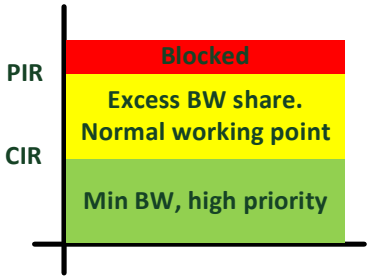


Figure 8-4. Per-Node CIR and PIR Levels

Figure 8-4 shows node bandwidth usage levels. The desired working point is the yellow area, where the node receives more than its minimum guaranteed bandwidth, shares excess bandwidth with its siblings, and does not reach its rate limit point.

The Tx-Scheduler tracks the bandwidth consumed by the nodes and manages the node state. Any non-blocked node that does not consume its configured minimum bandwidth is declared an **"Unsatisfied node"**. During arbitration flow, when selecting nodes from a sibling group, unsatisfied nodes receive higher priority to enable them to achieve the configured minimum bandwidth.

A Child Node cannot consume more bandwidth than allocated to its Parent Node. In the Tree topology configuration, the minimum guaranteed bandwidth must be fully provisioned. Hence, a node's minimum bandwidth must not be configured to be higher than the its own bandwidth limit or that of any of its ancestors all way to the Tree root. When configuring a minimum bandwidth for a node, it is also necessary to verify that its parent is able to receive this bandwidth as part of its minimum bandwidth guarantee setting.

Similarly, when multiple nodes are configured with minimum bandwidth, the sum of the minimum bandwidth of any sibling node group must not exceed that allocated for the parent. Each Tree Node can be assigned with an individual minimum bandwidth.

- **Excess Information Rate (EIR)** — Allows specifying a relative share of available bandwidth among Sibling Nodes. Sibling Nodes share a total bandwidth available from the Parent Node only after all the group's sibling nodes are in "satisfied" state (have received their minimum guaranteed bandwidth).

  The share of the Parent's bandwidth is relative, both with respect to the total amount of bandwidth available, and use of the bandwidth by other Sibling Nodes. If one of the Sibling Nodes cannot use its bandwidth share, the unused bandwidth can be utilized by other Sibling Nodes. "Bandwidth share" can be expressed in terms of weights or credits assigned to each tree Node.

- **Dual Rate Shaper** — A method that declares the bandwidth usage of a node. For a given Node, the Dual Rate Shaper configures all the following parameters: Bandwidth Limit (PIR), Minimum Bandwidth Guarantee (CIR), and its Excess Information Rate (EIR).

  In theory, "Unsatisfied Node" is a short transitional state of the node. As described in the CIR definition, the CIR setting must be fully provisioned to assure the Minimum bandwidth guarantee contract. Assuming this is true, the arbitration method between "Unsatisfied Nodes" can be any simple round-robin scheme, since the "Unsatisfied Nodes" node get higher priority to return to the "Satisfied" state.

  In some usage models, customers might prefer to configure the CIR to be Best Effort higher priority rather than a strict fully provisioned traffic. In practical terms, not all CIR users are really fully in use. Statistically, the bandwidth allocated for the CIR almost satisfies the real need. In some cases, more CIR user really try to use their CIR bandwidth, in this case the CIR users channels are oversubscribed. For these use cases, the above assumption that CIR traffic is always fully provisioned is true only in most cases, but not necessarily always. This means Nodes might be stacked in an "Unsatisfied State" for a longer period of time. To avoid bursty traffic in those cases, the E810's Tx-Scheduler also implements a WFQ algorithm with bandwidth allocation configuration for CIR nodes. Two sets of bandwidth allocation configuration parameters are declared for every node. One is used when the Node is in CIR state, and the other is used when it is in EIR state.

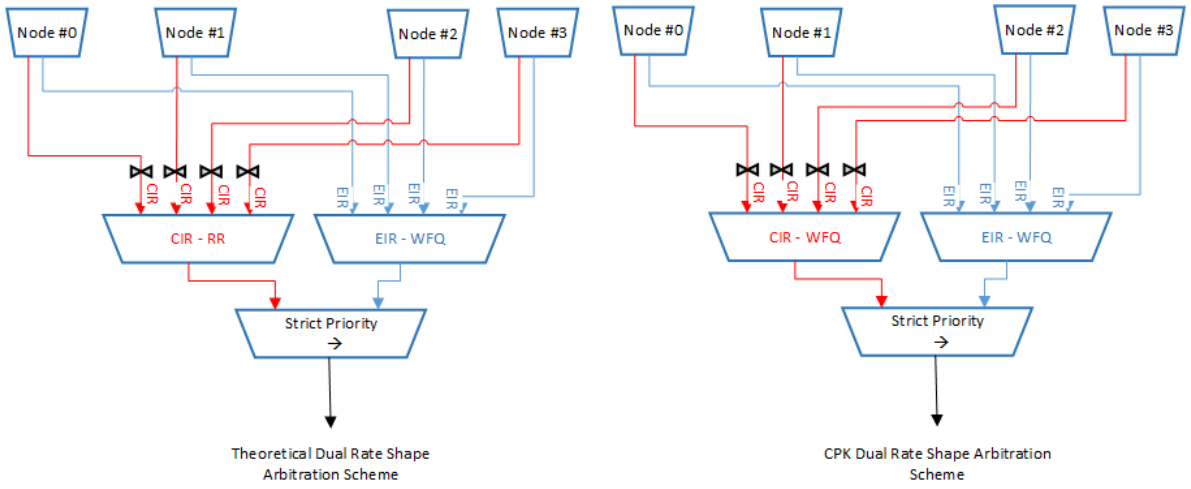


Figure 8-5. Dual Rate Shaper Arbitration Scheme

In Figure 8-5, the left scheme illustrates the usual implementation of Dual Rate Shaper Arbitration. Each node of the siblings group is connected to both EIR and CIR arbiters. The connection to the CIR arbiter is shaped to the CIR configuration of this node. A round-robin arbitration is implemented between all CIR nodes. Any CIR node has higher priority than the EIR nodes.

The right figure illustrates the E810's implementation. WFQ arbitration in both CIR and EIR. Implementation-wise, the Tx-Scheduler does not run two WFQ schemes in parallel. Instead, each of the eligible non-blocked sibling Nodes, sends its CIR value to the arbitration if it is in CIR state. Otherwise, it sends its EIR value. If at least one of the nodes is in CIR state, arbitration is performed among all CIR nodes. Otherwise, the arbitration is performed among all the available nodes.

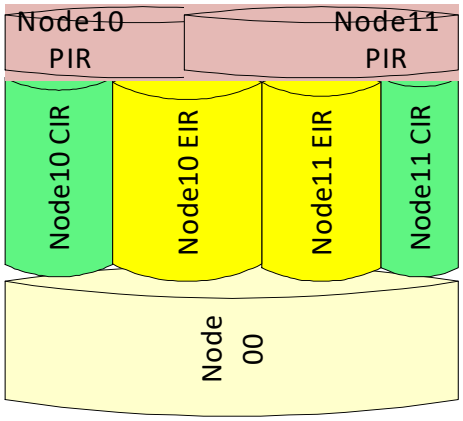


Figure 8-6. Parent Bandwidth Distribution Between Two Child Nodes

Figure 8-6 demonstrates the bandwidth distribution between two sibling nodes. The sum of CIRs of all siblings is less than or equal to the total bandwidth of the parent. The excess bandwidth is the part which is above the CIR for all siblings is shared between the siblings according the rate of each node. The PIR bandwidth of the nodes can overlap so the sum of the max bandwidth of all the siblings can be more than the total amount of parent's bandwidth.

Figure 8-7 demonstrates a wider picture of hierarchal bandwidth distribution.

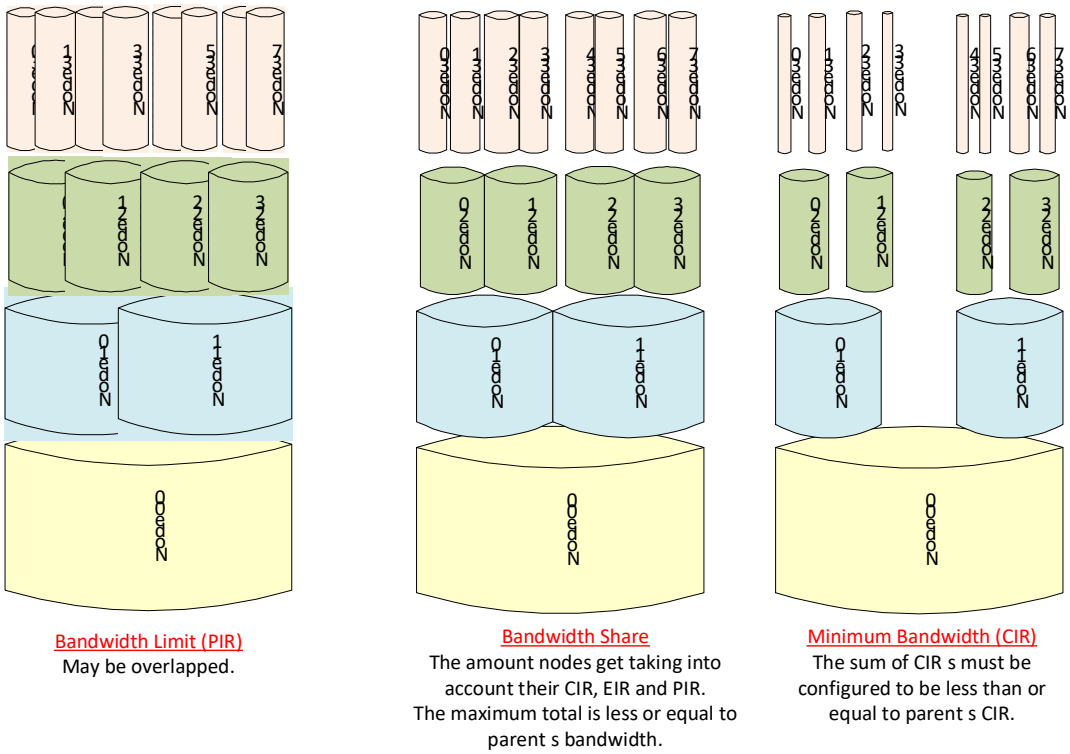


Figure 8-7. Hierarchal Bandwidth Distribution

8.3.2.3 Arbitration Schemes within a Sibling Group

The E810 supports several arbitration schemes that can be used to select a Node among Siblings in a hierarchal tree-based scheduling process. The selected node is called the group winner or "winner". This section explains the way the hierarchal tree-based scheduling process uses the selection or how the scheduling tree winner is selected.

Each set of Sibling Nodes has a configurable arbitration scheme assigned to it. Siblings participate in arbitration only if they are not blocked. Siblings can be configured to modify their arbitration scheme under different situations or in different states.

- **Strict Priority (SP)** — This arbitration scheme attempts to schedule Nodes based on their priority as long as the Nodes remain within their bandwidth limit.

  A Node can be pre-configured with constant priority, or can be configured to dynamically adopt a propagated priority from the previous layer. Another dynamic priority option is when a node or group of nodes is configured with minimum guaranteed bandwidth. In that case, every node whose current bandwidth consumption is below its configured minimum guaranteed bandwidth (node in unsatisfied state), is temporarily set with the higher priority. For more details on minimum bandwidth guarantee, see Section 8.3.2.2.3.

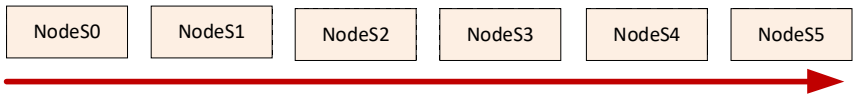


Figure 8-8. Arbitration Options - Strict Priority

- **Round Robin (RR)** — This arbitration scheme equally prioritizes Nodes, giving every Sibling Node a chance to be selected in cyclic round-robin fashion as long as it is not blocked.

  The arbiter iterates Sibling Nodes in cyclic round-robin fashion. In every scheduling cycle, the arbiter moves to the next Sibling Node and grants it another quanta (the quanta size for each node can be different). If the Node is blocked, the arbiter moves to the next Sibling Node. This arbitration allows even distribution of bandwidth among Sibling Nodes when all the Nodes are configured with the same quanta size.

  Note: This arbitration is not implemented in the E810 and is described here for background information.

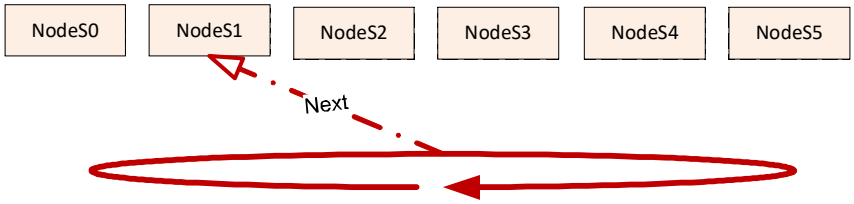


Figure 8-9. Arbitration Options - Round-Robin

- **Weighted Round-Robin (WRR)** — This arbitration scheme equally prioritizes Nodes and allows configuring each node with a different bandwidth by configuring each node with credits. The arbiter allows every Sibling Node a chance to be selected in cyclic round-robin fashion as long as it has credits and is not blocked. The node with most configured credits, survives more round-robin cycles and is granted greater bandwidth. When the credits of all siblings are exhausted, all the sibling nodes replenish their credits to the configured value.

  Note: This arbitration is not implemented in the E810 and is described here to clarify the WFQ description in the next paragraph.

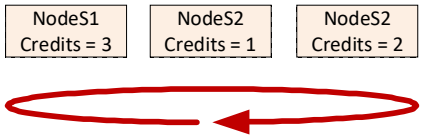


Figure 8-10. Weighted Round Robin Example Bandwidth Distribution: 50%, 33% and 17% for Nodes 1, 2 and 3 Correspondingly Credit Configuration is 3, 2 and 1

- **Deficit Weighted Round Robin (DWRR)** — This is an improved WRR scheme. In WRR, a quanta is granted for a node or queue without taking into account the actual size of the transmitted packet. A quanta almost never fits exactly to the packet boundaries. DWRR, however, is a responsive mechanism. After transmission, the delta between the quanta size and the actual transmission size is fed back to the Tx-Scheduler (Credit Update) and taken into account for the next scheduling decisions.

Note: DWRR is supported by previous NIC generations, but is **not implemented in the E810's Tx-Scheduler**. It is described here to provide clarification for the WFQ description in the next paragraph. Instead of DWRR, the E810 provides a better arbitration scheme called Weighted Fair Queuing (WFQ). WFQ is described in the next section.

- **Weighted Fair Queuing (WFQ)** — From the output perspective, WFQ arbitration appears similar to weighted bit-by-bit round-robin arbitration output. The transmitter does not interleave bits of different packets on the line, but when the last bit of a packet or a quanta is scheduled, the whole packet is scheduled. This point of time is called the departure time (DT) of the packet/quanta, and the scheduling grant flow for a packet/quanta transmission is called "dequeue". When a queue is selected for transmission, the DT of the winner is registered as the group DT call Last DT (LDT).

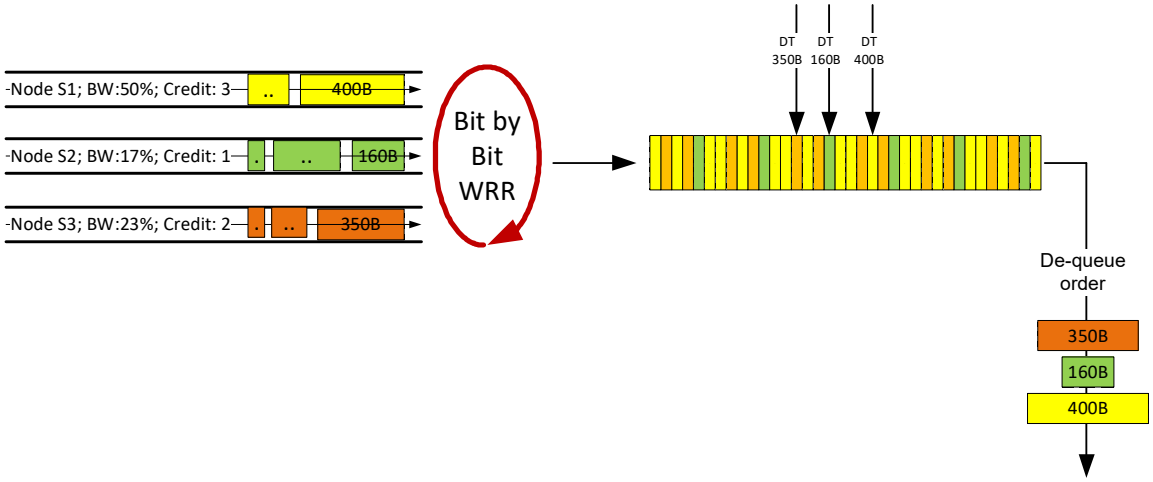


Figure 8-11. WFQ - Pseudo Bit-by-Bit WRR Arbitration

The WFQ arbitration method provides better (smaller) burstiness than DWRR. Its main advantages are:

— It takes in account the actual packet sizes in advance so less deficit update is required.

— Smaller packets naturally have a higher priority. Large packets never starve out small packets even momentarily.

As mentioned above, the Tx-Scheduler does not really run a bit-by-bit WRR for arbitration. Rather, it calculates the DT for each sibling node directly.

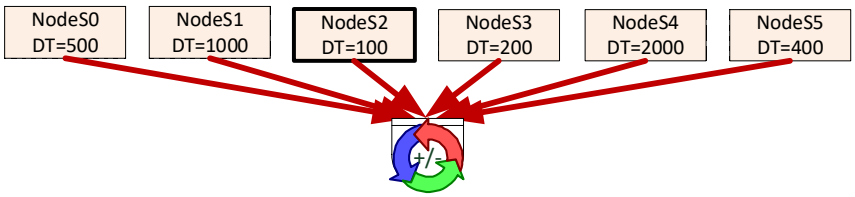


Figure 8-12. Arbitration Options - WFQ: Calculate DT for All Siblings and Pick the Lowest

The example in Figure 8-12 illustrates a sibling group of nodes in a WFQ arbitration. Node S2 is the winner in that particular example.

- **Combination of Strict Priority and Weighted Fair Queuing** — As described above, any node in any group can carry its own priority. This priority can be constant pre-configured, or dynamic. In each arbitration flow, the winning node is selected from the group's non-blocked siblings with the highest priority. Thus, different nodes can be configured with different priorities, even if they are siblings in the same group.

  Figure 8-13 shows two possible schemes of mixed arbitration. The basic rule is that each sibling group consists of a sub group with high priority nodes, a WFQ sub group with intermediate priority nodes, and a sub group with low priority nodes. In the lower example of Figure 8-13, NodeS0 is configured with the highest priority, NodeS1 with the next highest priority, NodeS2 and NodeS3 have the same priority, NodeS4 has a lower priority, and NodeS5 has the lowest priority in the group. In each sibling group, only one subgroup of nodes can carry one priority, all other nodes must carry a different priority each node. The upper example in Figure 8-13 is a subset of the general case.

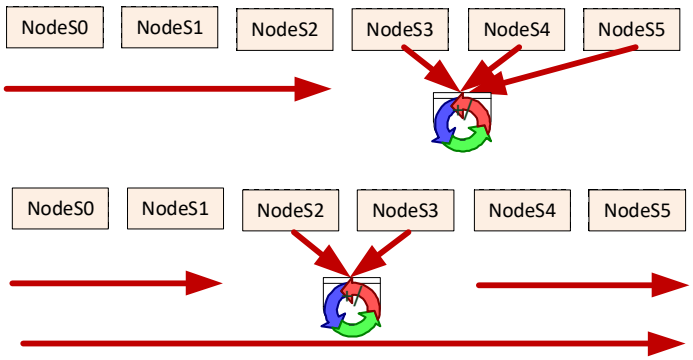


Figure 8-13. Combination of Strict Priority and Weighted Fair Queuing

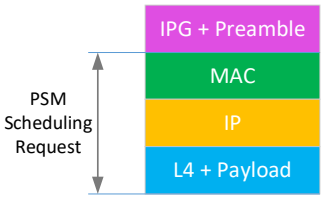
8.3.2.4 Credit Update

As described in the overview in Section 8.2.3.1, each stage of the Tx pipeline can receive more or less data on the actual Tx packet length, and can make a decision of packet length change or even drop the packet. Each of those changes are reflected as updates to the Tx-Scheduler. The report to the Tx-Scheduler is made as soon as the Tx-Pipe processing discover is required.

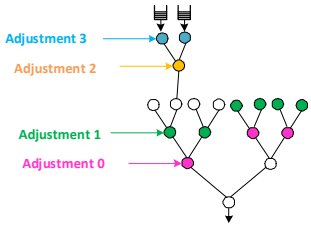
8.3.2.4.1 Four-Level Packet Adjustment Credit Update - Requirements

- Transmitted packets pass through different hierarchies in the network. The network hierarchies are reflected in the Tx-Scheduler layers structure. The capacity of each network unit is reflected in the rate limit and bandwidth allocation configuration of the nodes.
- Each unit in the network branch can deal with different portions of the packets.
- Typically, for the network hop, the whole packet including the IPG and the L1 header needs to be taken into account in the bandwidth calculation. Next hops, bridges, or the target station can consider a portion of the packet. For example, in tunneled environment, the target station does not see the whole packet, but only inner L4 part.
- The Tx-Scheduler is required to calculate up to four different adjustments of packets lengths for different Tx-Scheduler layers or nodes.
- The Tx-Scheduler is also required to calculate different adjustments of packets lengths for different types of packets.

- By default, the four adjustments point to:
  - Actual packet length on the wire including preamble and IPG.
  - Ethernet-layer (MAC to end of packet including padding. CRC is optional.
  - IP-layer (L3 IP header + L3 IP payload, inclusive) including ESP trailer.
  - TCP/UDP/… - layer (L4 header + L4 payload, inclusive).



- This way, the bandwidth consumption for different hops in the hops of the network is dependent on the actual portion of the packet run through this network unit.
- In the Tx-Scheduler, each nodes group is configured with the appropriate adjustment (2-bit setting). According to this configuration, each node is updated with the credits according the packets' format.
- When the credit update is propagated through the Tx-Scheduler pipeline and the tree structure, each node is updated with one of the four credit update values calculated as described above.



- This way, different adjustment can be configured per node in the scheduler.
- The credit update adjustment values should be programmable. Each COMMS customer (especially the Base station providers), has its own view on the network structure and restrictions.

8.3.2.4.2 Four-Level Packet Adjustment Credit Update - Implementation

As stated above, every transmitted packet is analyzed as part of the query flow to identify if it is transmitted, sent as loopback packet inside the platform, both, or dropped. In this query flow, the packet structure is analyzed by the E810's FlexiParser, described in Section 7.7

The E810 packet's FlexiParser, is a programmable packets analyzer. For each packet it analyzes, it generates up to 16 protocol reports. Each protocol report points to a packet field location inside the analyzed packet.

Protocol report is delivered to the E810 pipeline in the format:

- **{ProtocolID 8b; Offset 9b}**
  - ProtocolID = Identification of the protocol header found.
  - Offset = Octet offset from the beginning of the packet to the protocol header.

Packet Adjustment Calculator (PAC) is located after FlexiParser, it gets these 16 protocol reports and offsets (OCLs – Offset Collectors) per packet, and generates the 4 x length adjustments to the Scheduler. The PAC contains a table of 64 Adjustment Profiles (APs).

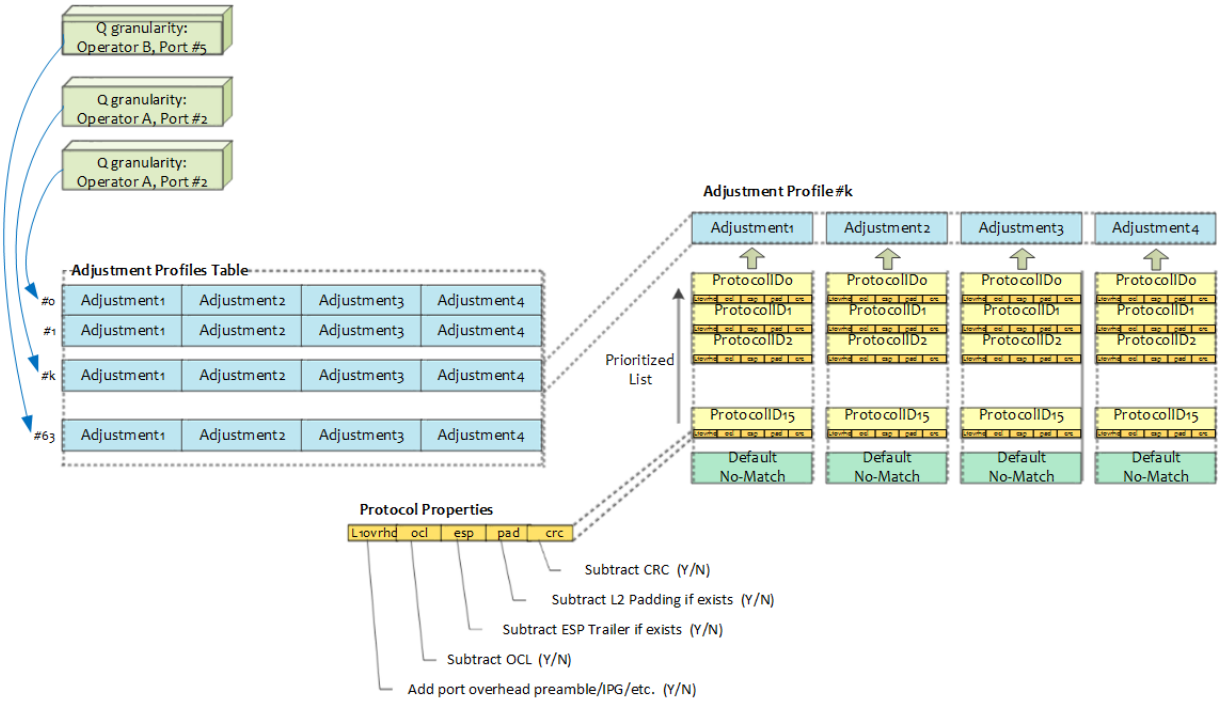


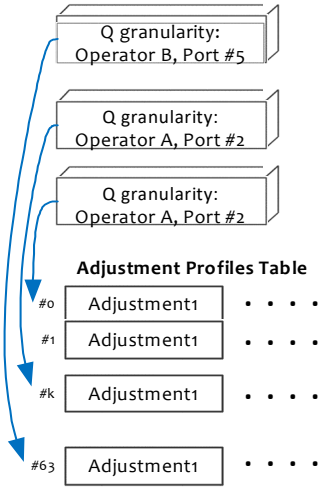
Figure 8-14. PAC Profile Structure

8.3.2.4.2.1 PAC - Adjustment Profile (AP) Association

Each AP is associated with a port and optionally with a Service Provider. This association is managed directly by PF. Per-PF allocation of APs is managed by software. The profile allocation is aligned with the FlexiParser root nodes allocation.

The FlexiParser gets the sourceVSI with the packet's header. The sourceVSI is used by the FlexiParser to determine the analyzing parse graph.

Each Tx-Queue is associated with an AP ID. This is configured in the Tx-Queue context by the PF. The AP ID is propagated with the packet's metadata and provided to PAC.



8.3.2.4.2.2 AP Profile Structure and Flow

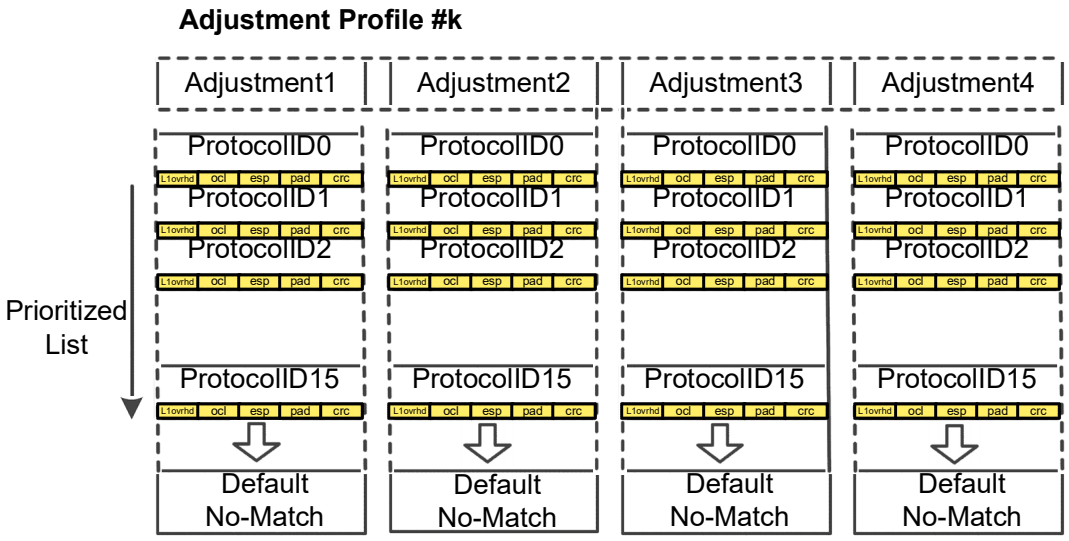


Figure 8-15. Adjustment Profile (AP) Structure

Each Adjustment Profile (AP) is made of 4 x adjustment components that generate the length adjustments. Each component defines a list of up to 16 x ProtocolIDs in prioritized order (Protocol ID 0 has the highest priority).

Each one of 16 protocol IDs FlexiParser provided per each packet, are compared with the list of the protocol ID lists of the relevant AP. The comparison is priority-based. Per adjustment, when a ProtocolID N matches a FlexiParser output protocol ID, The next protocol IDs in this list are ignored.

The offset of the matched protocol is captured in an Offset Collector (OCL) for calculating the length adjustment. The selected OCLs [1...4] are used for packet length adjustments [1...4], which are sent as update to the Tx-Scheduler in Update #2 flow.

Per each adjustment, if there is no match between the FlexiParser outputs and the AP's Protocol IDs, a hard-coded 0x1FF value is used as an adjustment value.

8.3.2.4.2.2.1 Adjustment Profiles Properties

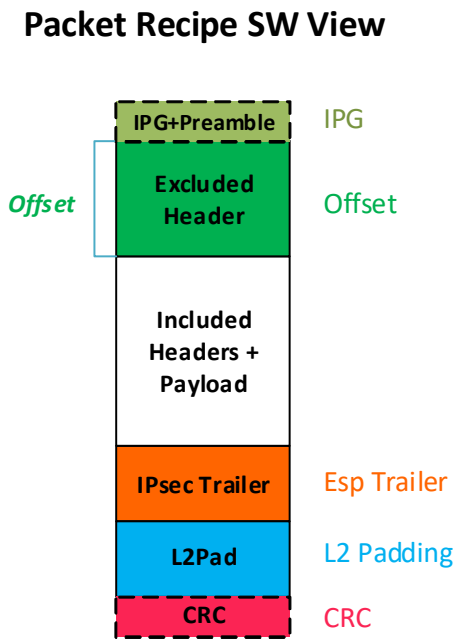


Figure 8-16. Adjustment Profile Properties (APP)

Each protocol ID of the AP is configured with a set of five flags. Those flags define the way this adjustment is calculated.

- The "CRC" flag marks if packet's CRC field is included in packet length for scheduling purposes.
- The "PAD" flag marks if in case of small packets. The 64B padding is included in packet length for scheduling purposes.
- The "OCL" flag marks if for this protocol ID. All headers until this header are included in packet length for scheduling purposes.
- The "L1ovrhd" flag marks if the L1 preamble and the IPG is included in packet length for scheduling purposes.

8.3.2.4.2.3 Packets Adjustment Examples

Example #1: Adjust for True L1 Length on the Wire:

- Adjustment #1 =

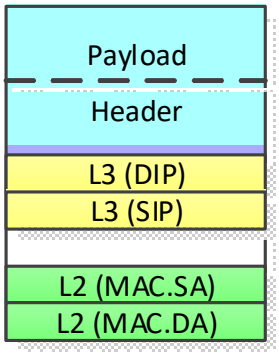
Flag	Operation Value	Description
L1ovrhd	1	Overhead Preamble/IPG = 20.
OCL	1	OCL not relevant, consider start of packet offset.
PacketData	Pkt Data	PacketData beyond OCL.
ESP	1	Do not subtract ESP trailer.
L2PAD	1	Do not subtract ESP padding.
CRC	1	Do not subtract ESP CRC field.

Packet length for Scheduling = Scheduled packet length + IPG preamble + CRC

Example #2: Adjust for L3 Length:

- Adjustment #3 =

Flag	Operation Value	Description
L1ovrhd	0	Overhead Preamble/IPG = 20.
OCL	0	OCL Outer IP header offset
PacketData	Pkt Data	PacketData beyond OCL.
ESP	1	Do not subtract ESP trailer.
PAD	0	Do not subtract padding.
CRC	0	Subtract CRC field.



Packet length for Scheduling = Scheduled_Pkt_Length – OCL (IP) – L2PAD

8.3.2.4.3 Four-Level Packet Adjustment Credit Update - Programming Registers

The configuration of the four adjustment functions are listed below. This list includes the adjustment calculation based on the parser output. The parser configuration is part of the parser chapter.

- GL_TDPU_PSM_DEFAULT_RECIPE
- GL_TDPU_PSM_PE_PROF_ID
- GLTDPU_NEG_PSM_CREDIT_UPDATE_EN

8.3.3 Network Topologies in Tx-Scheduler

The E810 supports various network topologies and various usage model targets. Each topology is reflected in the Tx-Scheduler topology. The E810 can work as a single-port or a multi-port device. At initialization, EMP reads basic configuration from the NVM and configures the Tx-Scheduler with an initial configuration. The parameters read from the NVM (words Logical Layer Config and Logical Layer Structure) are number of ports, and number of scheduling layers (5, 7, or 9).

The E810 supports up to 16K LAN Tx-Queue. For RDMA, The E810 supports up to 256K Queue Pairs (QPs). The 256K QPs of RDMA are organized in Queue Sets (QSets). Each QSet represents a group of QPs belonging to a function/TC. Typically, the number of RDMA QSets is the number of RDMA-enabled functions multiplied by the number of supported TCs. For scheduling purposes, a QSet is considered as one scheduling unit with its configuration for priority, bandwidth allocation, and limit. When the Tx-Scheduler schedules a quanta to a RDMA QSet, the PE shares this quanta between the QPs belonging to the QSet in a Round Robin manner.

The Tx-Scheduler supports up to 16K leaf nodes. Each leaf node can be associated with one LAN Tx-Queue or one RDMA QSet.

As mentioned above, the number of scheduling layers is a parameter loaded from the NVM, and can take one of the following values: 5, 7, 9. As the number of layers is lower, the performance of the Tx-Scheduler is higher since each decision flow covers fewer layers.

Table 8-23. Tx-Scheduler Performance as Function of Number of Active Layers

Topology Depth	# Clock Cycles / Quanta per Second	Quanta Size @ 100G Part
9 layers	24 cc / 18.58 M	673B
7 layers	20 cc / 22.3 M	561B
5 layers	16 cc / 27.9 M	448B

Note: In the first generation of the E810, the options for 5 and 7 layers are off POR.

When operating in 7 layer mode, Layers 5 and 7 are skipped. All other layers are fully functional.

8.3.3.1 ETS-Based Scheduler Configuration

Figure 8-17 shows a logical diagram describing the ETS-based bandwidth distribution scheme.

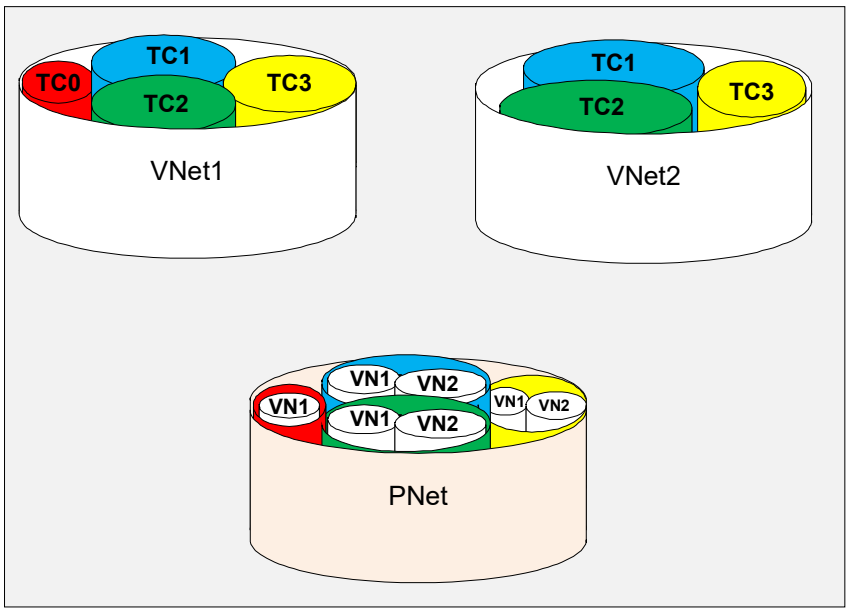


Figure 8-17. ETS-Based Bandwidth Distribution

This figure shows two Virtual Networks VNet1 and VNet2. Each virtual network carries traffic of different types by Traffic Class (TC). Both virtual networks merge into the Physical Network. The physical network is configured accordingly to ETS specification. The bandwidth of the physical network is divided between TCs. Each TC carries traffic belonging to respective traffic types of both virtual networks. The bandwidth distribution between virtual networks within same TC is invisible outside of the chip.

The diagram of the physical network shows the concept of bandwidth distribution in an ETS-based configuration. Bandwidth is first distributed between types of traffic (or Traffic Classes) and then distributed between virtual networks within each TC. Therefore, if one of the virtual networks does not have traffic of the certain type currently available, the remaining bandwidth is consumed by the same traffic type from other virtual networks. For example, if VNet2.TC3 does not have any work to perform, the remaining TC3 bandwidth is consumed by VNet1.TC3, as long as it has enough work to occupy it, and as long as it does not exceed any bandwidth limit.

This scheme allows bandwidth allocation per traffic type for each scheduling component and VSI, and does not support a bandwidth allocation per virtual network.

This is the default bandwidth allocation scheme for in DCB networks.

Figure 8-18 shows an example of system configuration and bandwidth allocation using the ETS-based scheduler configuration scheme.

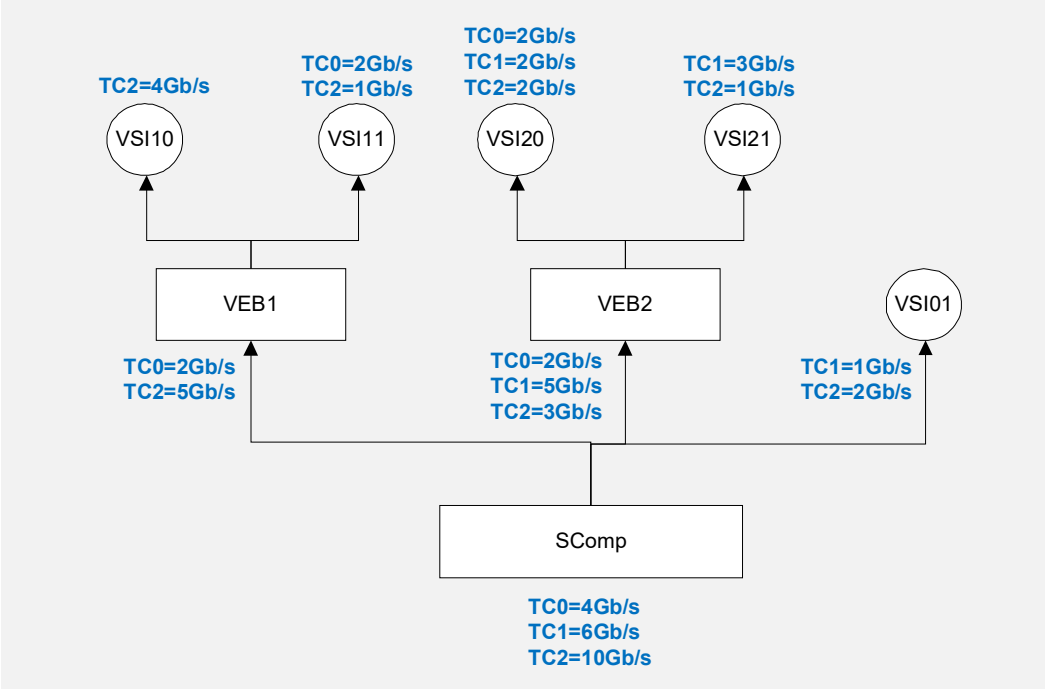


Figure 8-18. Example of ETS-Based System Configuration

Each of the Scheduling components (SComp, VEB1 and VEB2) and VSIs can be configured with ETS (bandwidth allocation per traffic type or TC) and bandwidth limit. Bandwidth allocation must be consistent. That is, if TC has a certain bandwidth allocated on the egress SComp port, the sum of the bandwidth allocated to the same TC on each level of tree hierarchy should be equal to the bandwidth allocated on SComp egress port. For example, if:

SComp.TC0 = 4 Gb/s

then:

VEB1.TC0 + VEB2.TC0 + VSI01.TC0 = 4Gb/s

and:

VSI10.TC0 + VSI11.TC0 = VEB1.TC0

Figure 8-19 shows an ETS-based scheduling tree configuration corresponding to the system configuration shown in Figure 8-18.

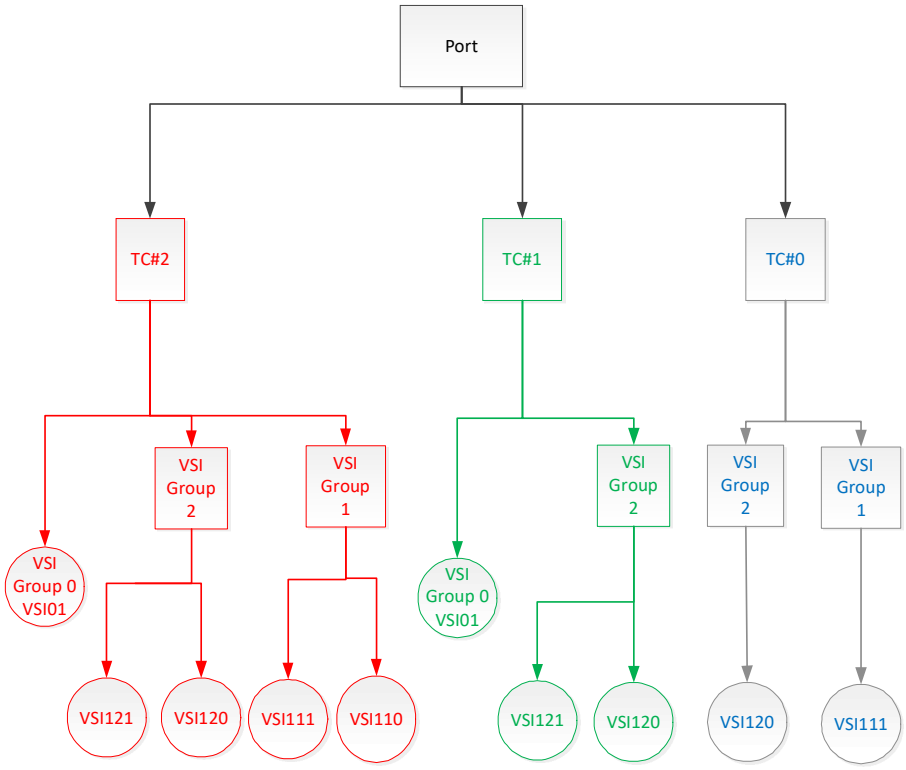


Figure 8-19. ETS-Based Scheduling Tree Configuration Example

The structure of this scheduling tree resembles the topology of the system configuration shown in Figure 8-18. This tree has four tree levels (one not shown):

- **Port Level** — This is the level that arbitrates between the physical ports.
- **TC Level** — This is the lowest tree level above the physical port. It is configured with physical port or SComp egress port ETS bandwidth allocation and priority (for example, nodes TC0, TC1 and TC2 in Figure 8-19).
- **Scheduling Component Level** — The next scheduling tree level. This level consists of Scheduling Components (VEB/PA) and VSIs directly attached to SComp. Each tree node corresponds to the TC/UP configured for the Scheduling Component/VSI. A single scheduling component and VSI can appear in the tree multiple times, once for each instance configured as TC/UP (for example, nodes VEB1_TC0, VEB1_TC2, VEB2_TC0, VEB2_TC1, VEB2_TC2, VSI01_TC1 and VSI01_TC2 in Figure 8-19).
- **VSI Level** — Last tree scheduling level. This level consists of the VSIs of scheduling components. Each tree node corresponds to the TC configured for the VSI. A single VSI can appear in the tree multiple times, once for each configured TC/UP (for example, VSI10_TC2, VSI11_TC0, VSI11_TC2, VSI20_TC0, VSI20_TC1, VSI02_TC2, VSI21_TC1 and VSI21_TC2).

Internal Scheduling Tree configuration is not visible to software. The EMP firmware creates and manages the internal scheduler configuration tree and translates software controls to the actual scheduler configuration. For example, when software provides an ETS configuration of VEB1, it lists TCs enabled for VEB1 (TC0 and TC2), and specifies an ETS bandwidth allocation per TC. The EMP firmware allocates multiple scheduling nodes for VEB (VEB1_TC0 and VEB1_TC2) and configures those nodes with credits provided by software.

Since the ETS configuration of VEB results in the creation of multiple tree nodes (one per TC), and those nodes reside on different subtrees, the credits provided should be relative to other VEBs/VSIs belonging to the same TC, and not relative to the TCs enabled for VEB/VSI. For example, VEB1_TC0 credits must be relative to VEB2_TC0 credits, and not to VEB1_TC2 credits.

Software can enable a bandwidth limit for any scheduling component and VSI. If a scheduling component or VSI has more than single traffic class enabled, it appears multiple instances in the Transmit Scheduling Tree. Use of shared or basic bandwidth limits is hidden from the software, and the decision whether to use basic or shared bandwidth limits is taken by EMP firmware depending on the selected bandwidth allocation mode and system configuration.

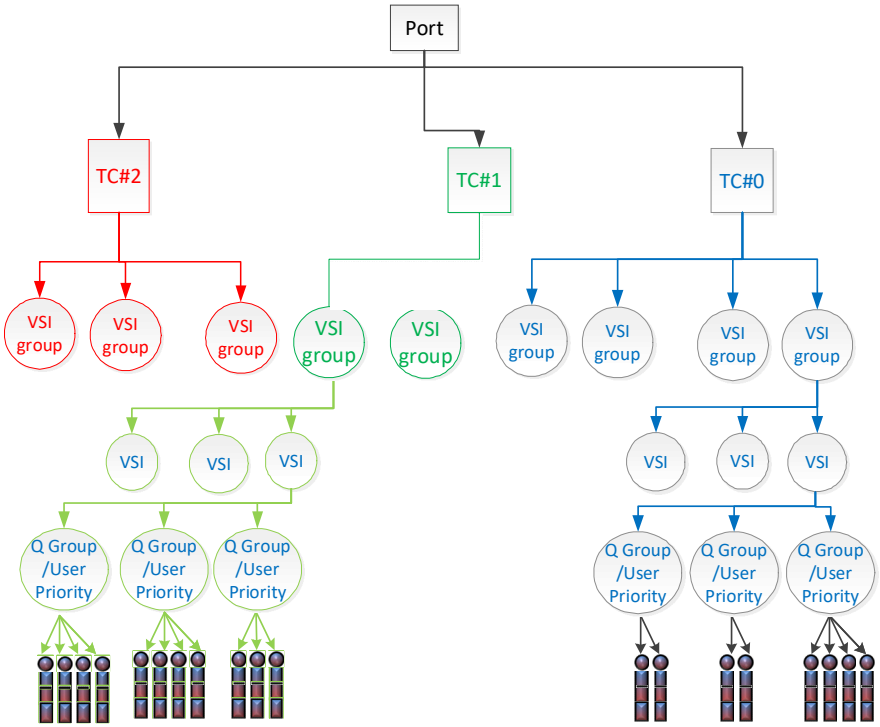


Figure 8-20. ETS-Based Scheduling Super Set Tree Configuration Example

Figure 8-20 illustrates an example of a super set topology for a DCB network, mainly used in the Cloud/ Enterprise market as detailed below:

- **Port Level** — This is the level that arbitrates between the physical ports.
- **TC Level** — This is the lowest tree level above of the physical port. It is configured with physical port or SComp egress port ETS bandwidth allocation and priority (for example, nodes TC0, TC1 and TC2 in Figure 8-20).
- **Scheduling Component Level** — The next scheduling tree level. This level consists of Scheduling Components (VEB/PA) and VSIs directly attached to SComp. Each tree node corresponds to the TC/ UP configured for the Scheduling Component/VSI. A single scheduling component and VSI can appear in the tree multiple times, once for each configured TC/UP (for example, nodes VEB1_TC0, VEB1_TC2, VEB2_TC0, VEB2_TC1, VEB2_TC2, VSI01_TC1 and VSI01_TC2 in Figure 8-20).
- **VSI Group Level** — This level consists of the groups of VSIs scheduling components. Some customers can use this intermediate level to bunch multiple VSIs into a scheduling groups. Each tree node corresponds to the TC configured for the VSI group.

- **VSI Level** — This level consists of the VSIs of scheduling components inside the VSI group. Each tree node corresponds to the TC configured for the VSI. A single VSI can appear in the tree multiple times, once for each configured TC (for example, VSI10_TC2, VSI11_TC0, VSI11_TC2, VSI20_TC0, VSI20_TC1, VSI02_TC2, VSI21_TC1 and VSI21_TC2).
- **Queue Group Level** — This level consists of the groups of queues inside VSI scheduling components. Some customers can use this intermediate level to bunch multiple Tx-Queues into scheduling groups. The queue groups of a VSI can be configured with the same priority or with different scheduling priorities. The queue group might act as a User Priority Level.
- **Queue Level** — This level consists of Queue Scheduling Components. Each Tx-Queue or RDMA QSet is scheduled individually.

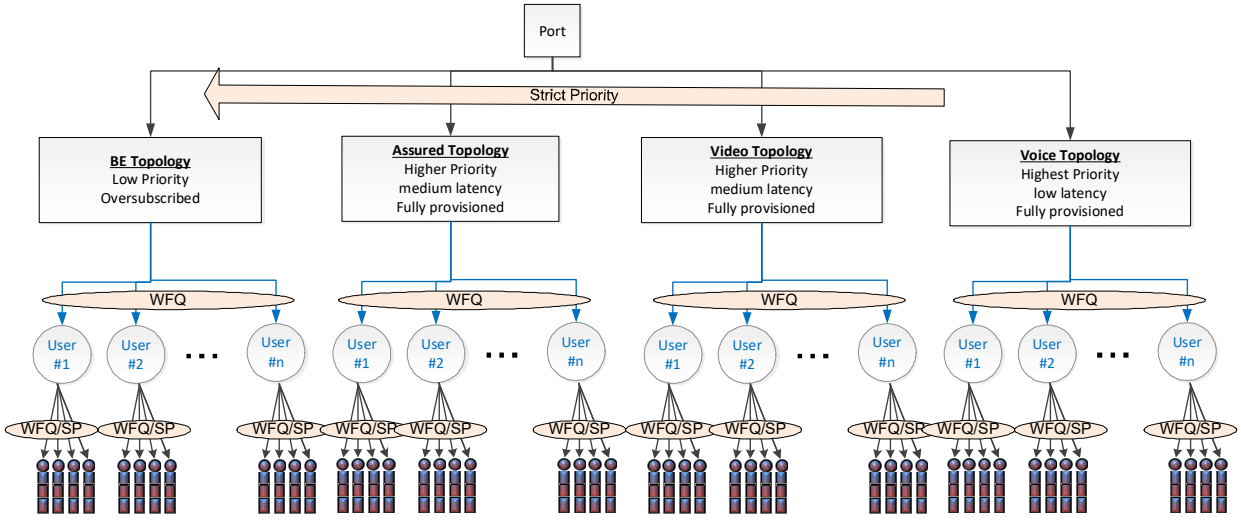


Figure 8-21. ETS-Based Scheduling Tree in BRAS Example

Figure 8-21 shows an example for BRAS scheduling tree in ETS-based topology mode.

- **Port Level** — This is the level which arbitrates between the physical ports.
- **Traffic Type/Priority Level** — This is the lowest tree level above the physical port. It is configured with physical port ETS bandwidth allocation and priority. Usually in the BRAS usage model, this layer is configured with Strict priority. The Voice Traffic with the highest priority, followed by Video Traffic, Assured Traffic, and Best Effort traffic with the lowest priority. All traffic types are fully provisioned and rate-limited to their provisioned rate.
- **User Level** — In ETS-based topology, this Layer arbitrates between the users/subscribers separately in each traffic type subtree.
  - In Voice, Video, and Assured Traffic types, the user nodes are rate limited to their provision rate. Since the parent node is also fully provisioned, there is no real congestion in the user layer for those traffic types.
  - In the Best Effort traffic type there is real congestion and scheduling between the users. This traffic is usually oversubscribed. Usually in COMMS application three types (or SLA levels) of users are served. The distinction between user levels can be implemented by dedicating minimum bandwidth allocation for high level users. Allocating higher weight bandwidth allocation. The rate limit configuration of user node also depends on user level.

- **Queue Level** — This level consists of Queue Scheduling Components. Each Tx-Queue is scheduled individually. Queue bandwidth configuration can be evenly shared between the user queues or setting different SLA per queues of the same level can be used for providing different levels of service for the queue user (for example, when the user is called by the Skype application, he prefers to give higher bandwidth/priority to this traffic inside the allocated bandwidth share). Thus, queue level can also be used for user level priority.

8.3.3.1.1 Shared RL in ETS-Based Scheduler Configuration

When building an ETS-based topology, Shared Rate Limiter (SRL) must be available. The SRL is used to limit the total amount of bandwidth consumed by a specific user (in BRAS case) or by a VF (cloud/ enterprise case).

8.3.3.2 VNet-Based Scheduler Configuration

Figure 8-22 shows a logical depiction of a VNet-based bandwidth distribution scheme.

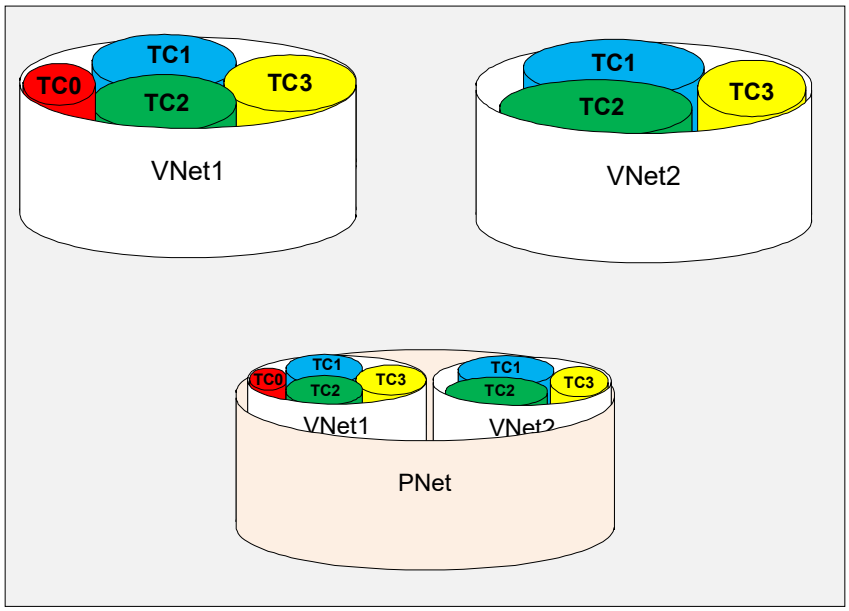


Figure 8-22. VNet-Based Bandwidth Distribution

Each of the two Virtual Networks VNet1 and VNet2 carries traffic of different types according to Traffic Class (TC). Both virtual networks are merged into the Physical Network, which is configured to distribute the available bandwidth between virtual networks first, then within each virtual network according to traffic types configured for that virtual network.

This bandwidth distribution scheme is logical for cloud-based environments with multiple tenants willing to share physical resources, and provides isolation of bandwidth allocation throughout the fabric. This scheme is also used in the COMMS market by some of the service providers.

This scheme also enables distribution of bandwidth between virtual networks, and provides global SLA guaranty without being limited to particular traffic type, as is the case with ETS-based schemes. Bandwidth can be then allocated according to traffic type within each virtual network. For example, if there is not enough demand to use all the resources of VNet2.TC3, the remaining bandwidth can be consumed by other traffic types configured for VNet2 (VNet2.TC1 and VNet2.TC2), as long as there is demand and no bandwidth limit is exceeded, rather than VNet1.TC3 consuming the unused bandwidth, as would happen in an ETS-based scheme.

Figure 8-23 shows an example of the system configuration and bandwidth allocation using a VNet-based scheduler configuration scheme.

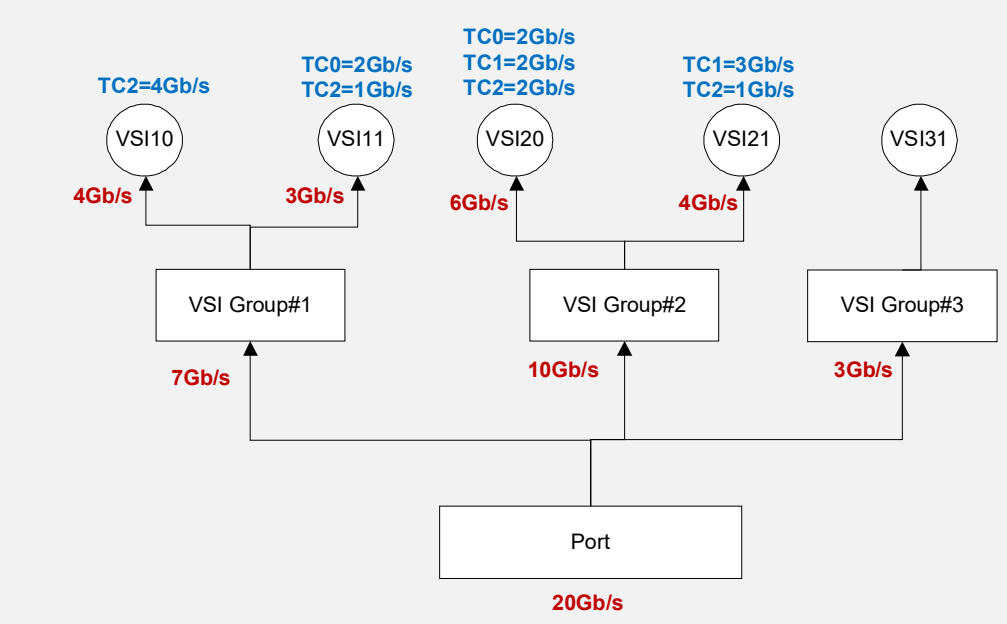


Figure 8-23. Example of VNet-Based System Configuration

Each of the scheduling components and VSIs has an allocated bandwidth and can have an associated bandwidth limit. VSIs are also configured with ETS, or per traffic type bandwidth allocation. Bandwidth allocation must be consistent. The bandwidth allocated to the scheduling component should be equal to the sum of bandwidth allocated to all VSIs, and the bandwidth allocated for VSI must be equal to the sum of bandwidth allocated to all the TCs. For example, the bandwidth allocated to VSI10 (4 Gb/s) and VSI11 (3 Gb/s) should be equal to the bandwidth allocated to PF#1 (7 Gb/s).

Figure 8-24 below shows a VNet-based scheduling tree configuration corresponding to the system configuration described in Figure 8-23.

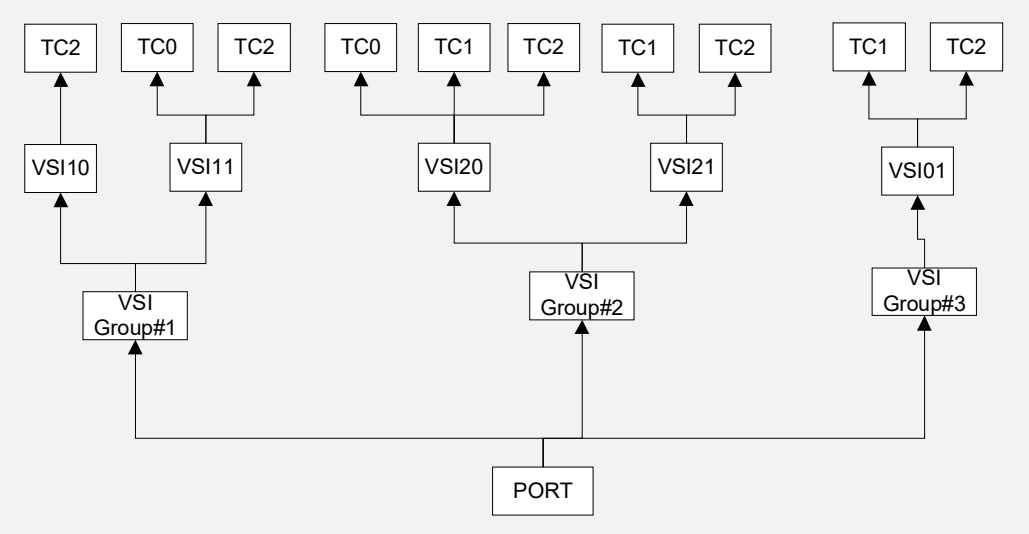


Figure 8-24. VNet-Based Scheduling Tree Configuration Example

The software is not aware of the internal scheduler configuration tree structure and uses a logical representation (shown in Figure 8-24) to program scheduler configuration tables. EMP firmware, translates software Admin Queue commands, and performs the actual configuration of the scheduler configuration tables.

If the software needs to instantiate a bandwidth limit for the scheduling component or VSI, EMP firmware should use a basic bandwidth limit, since each scheduling component and VSI is represented by a single tree node.

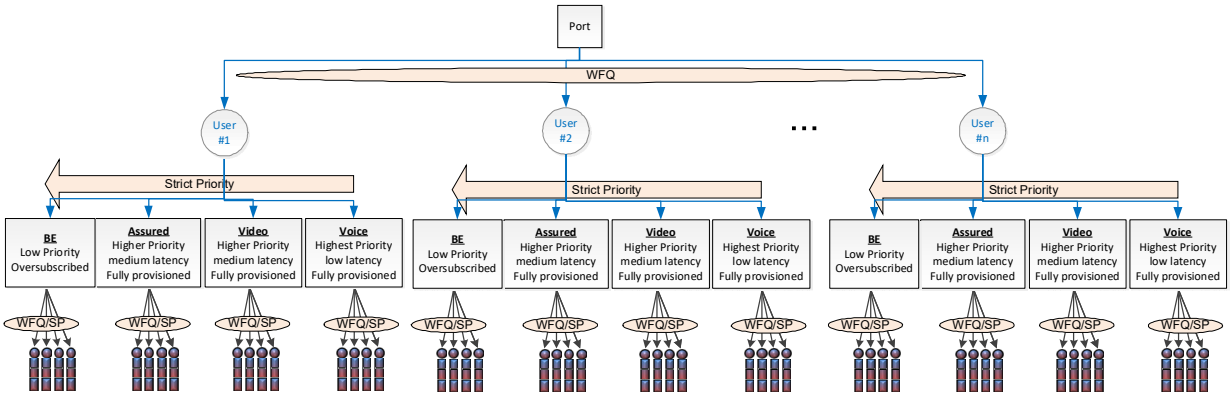


Figure 8-25. VNet-Based Scheduling Tree in BRAS Example

Figure 8-25 shows an example of a VNet-based topology usage in a BRAS network.

- **Port Level** — This is the level that arbitrates between the physical ports.
- **User Level** — In VNet-based topology, port bandwidth is shared among users according to their SLA, as follows:
  - In Voice, Video and Assured Traffic types, the user nodes are rate-limited according to their provision rate. Since the parent node is also fully provisioned, there is no real congestion in the user layer for those traffic types.
  - In the Best Effort traffic type, there is significant congestion and scheduling between the users, as this traffic is usually oversubscribed. Usually in COMMS applications, three types (or SLA levels) of users are served. The distinction between user levels can be implemented by setting a minimum bandwidth allocation for high level users, this allocating higher weight bandwidth allocation. The Rate limit configuration of user nodes also depends on user level.
- **Traffic Type/Priority Level** — In each user's subtree, bandwidth is shared between the four traffic types. Usually in the BRAS usage model, this layer is configured with strict priority: Voice Traffic with the highest priority, followed by Video Traffic, Assured Traffic and Best Effort traffic with the lowest priority. All traffic types are fully provisioned and rate limited according to their provisioned rate.
- **Queue Level** — This level consists of Queue Scheduling Components. Each Tx-Queue is scheduled individually. Queue bandwidth configuration can be evenly shared between the user queues or can be set to different SLAs so that queues of the same level can be used for providing different levels of service to the queue user. (For example, when a user calls by the Skype application, it is desirable to set a higher bandwidth/priority to this traffic in the relevant bandwidth share). In this way, queue level can also be used for user level priority.

8.3.3.3 VNET and ETS Topologies Support by One General Purpose Tree Structure in the E810

The previous products (like the X710/XXV710/XL710) limit the scheduling structure to ETS-based topology. In this product generation, the ETS vs. VNet support was a product mode. The firmware in these products supports ETS-based topology and was sufficient for enterprise usage models in DCB networks.

In the E810, both modes are supported. There is no need to set a mode. When DCB is enabled, TC level is added to the root of the tree, so ETS is always supported. The PF is allowed to configure node priority in any layer of the tree it owns. Thus, user priority is always supported.

Therefore, without mode changing, both modes are opened for PF programming by configuring the TC layer with priority or by setting the leaf layer with different priorities. This is a mixed mode where both TC level priority and user level priority are also supported per PF configuration.

A simple way to have a VNet topology support, is to enable only one of the TCs in layer 2 and leave the user-level priority active.

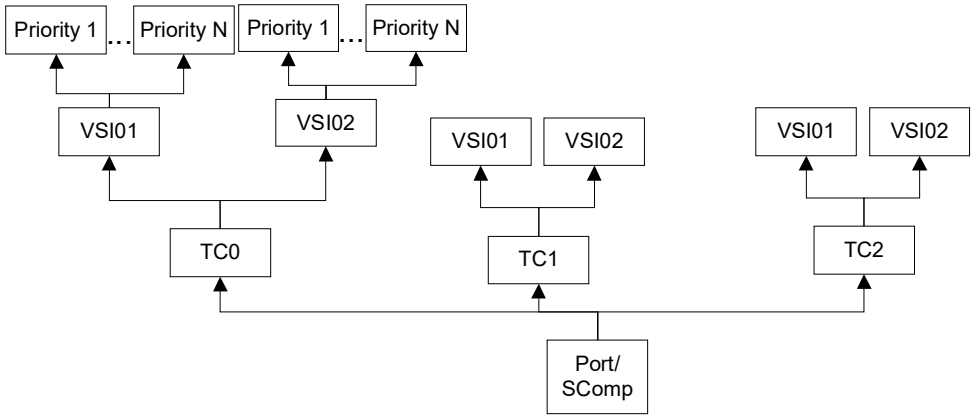


Figure 8-26. Mixed Topology VNet +ETS Example

8.3.3.4 Tx-Scheduler Configuration Process

The Scheduler implements one set of configuration tables shared by all PCIe functions. To synchronize access to the shared configuration table and maintain configuration consistency, direct access to the internal scheduler configuration is restricted to EMP.

The Scheduler does not expose its internal configuration tables to the standard deployment software. A set of registers allowing indirect access to the internal scheduler structures is exposed to firmware to allow programming of the Scheduler configuration tables. These registers are exposed in debug mode to other privileged software. A default configuration of the scheduling tables is performed by EMP firmware and based on the internal scheduling configuration profiles kept in NVRAM. NVRAM data includes the number of ports, number of scheduling layers (5, 7, or 9).

A physical function driver is considered to be trusted software and is allowed to modify the default configuration by proxying its requests through firmware using Admin Queue commands (see Section 8.3.4.3.6). Software should use an Admin Queue interface to communicate its scheduler configuration requests with EMP. EMP is responsible for constraining access to the scheduling configuration tables based on the software privilege level, shared resources, management mandatory, and resource ownership.

Most of the scheduling components are shared among physical functions. Once a scheduling component is allocated with a function, it is owned by a single physical function. If a scheduling component is shared among physical functions (for example, DCB bandwidth configuration), this component is configured and owned by the DCB agent running in EMP firmware.

8.3.3.4.1 Tx-Scheduler Configuration and Management Principles

- At initialization, a default topology is built, providing a subtree root for each PF. More details are provided in Section 8.3.4.1.
- The PF is able to build the required topology inside its subtree.
- EMP serves the PF in its task of Tx-Scheduler configuration and enforces basic topology rules. For example:
  - — Prevents the PF from affecting the behavior and performance of other PFs.
  - — Limits sibling groups size to NVM setting per logical layer.

  - — Does not let software remove nodes that have children.
- Some examples for rules that are not enforced by EMP firmware, but that PF software is responsible for keeping:
  - — TC or node priority consistency between nodes and layers.
  - — PPS/BPS group consistency.
  - — Tx-Queue and leaf node priority setting.
- A minimal set of scheduling components is used:
  - — **Port** — For each LAN port, a port node is allocated. This point reflects the physical port bandwidth.
  - — **TC** — This layer supports the DCB operation if it is active. The TC nodes are connected as child objects nested under the port node. When the DCBX message changes the DCB setting, this is reflected in this layer. The EMP adds TC nodes according the DCBX parameters. The bandwidth management for the TC layer is also configured by the EMP firmware according the DCBX parameters. After setting the TC layer, the PFs are notified and adjust their subtree with the new DCB setting. At init time, prior to receiving the DCBX message, TC0 is active and the entire topology is built on it. In this way, when further TCs are required, the tree layer remains unchanged.
  - — **VSI** — VSI is a scheduling component. It can reflect VM, VF, or PF.
  - — **LAN Tx-Queue/RDMA QSet** — This is a Tx-Scheduler leaf node. A LAN Tx-Queue or a PE QSet is connected to the leaf node.
  - — **Aggregator** — Additional general purpose scheduling component (called aggregator).
    - Aggregator can be added and located anywhere in the tree.
    - The aggregator is used to build subgroups of scheduling components when it is required. The subgroup can be configured with bandwidth allocation, minimum bandwidth, and limit.
- Queue can be connected to any Tx-Scheduler leaf node.
- When DCBX adds or removes TCs:
  - — The port's TC is expanded.
  - — PF builds the topology connected to the enabled PF/TC nodes. It can move existing subtrees and Tx-Queues, or add/delete scheduling components.
- Section 8.3.3 provides some examples of typical tree topologies for different usage models illustrating the usage of all scheduling components as well the topology building flows.

8.3.4 Flows

8.3.4.1 Tx-Scheduler Initialization Flow

As part of the device initialization flow, the Tx-Scheduler is initialized in two main steps:

- Hardware initialization.
- EMP basic topology configuration.

8.3.4.1.1 Hardware Initialization

- All the CSRs of the Tx-Scheduler block are reset to their default values and then auto-loaded according NVM settings.
- All Tx-Scheduler tables are cleared (all fields a zeroed).

8.3.4.1.2 EMP Initial Topology Configuration

For each PF, EMP allocates one Port node with one VSI node connected to it. If DCB is supported (NVM configuration), a TC layer is allocated as layer #2 topology. This VSI is the initial VSI of this PF. At a later stage, as part of the PF driver init flow, one initial queue is allocated to each initial VSI. Like for every VSI, PF can add more queues to the VSI. It can also organize the queues belonging to a VSI in groups according to bandwidth, with or without QoS (user Level TC or just queue groups) configuration.

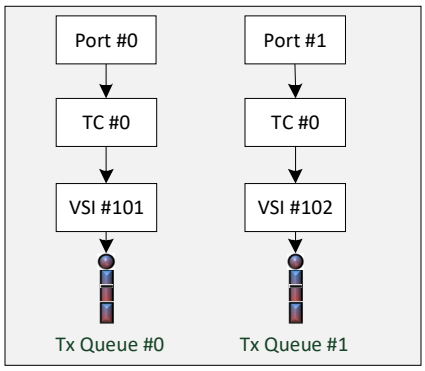


Figure 8-27. Example of Initial Configuration of a Dual Port Device

8.3.4.2 Resets Flows

Tx-Scheduler responds differently to different reset signal types. The following paragraphs detail the expected behavior for each reset type.

8.3.4.2.1 Power On Reset (POR), Core Reset (CORER), or Global Reset (GLOBR)

All data in the Tx-Scheduler tables are cleared. EMP configures all Tx-Scheduler tables according to the NVM setting of the device mode. For more details, see Section 8.3.3.4 and Section 8.3.4.1.

8.3.4.2.2 EMP Reset (EMPR)

EMP reset followed by GLOBR. Tx-Scheduler is reset according the CORER flow detailed in Section 8.3.4.2.1.

As part of the GLOBR reset as shown above, Tx-Scheduler registers and tables are reset and the EMP reconfigures the initial topology. Refer to Section 8.3.3.4 and Section 8.3.4.1 for details.

8.3.4.2.3 Function-Level Reset (FLR), PF Reset (PFR), VM Reset (VMR), VF Reset (VFR), or VFLR

Function-level reset events are not directly propagated to the Tx-Scheduler and no activity is expected in hardware for those events.

As part of the function-level reset flows, depends on the Reset Type the PF or the EMP firmware is responsible for function resources cleanup. The cleanup flow includes Tx-Scheduler resources cleanup using Delete Element(s) AQ Command detailed in Section 8.3.4.3.6.12.

8.3.4.2.4 PCIe Reset (PERST) or In-Band Reset (PCIR)

8.3.4.2.4.1 Single-Home Device

PCIe Reset in a Single-Home topology is followed by GLOBE and/or CORR (depending on the *veto* bit).

8.3.4.3 Tx-Scheduler Configuration Process

This section describes how to configure Scheduling configuration principles, and guides firmware through implementation of the Admin Queue commands defined in Section 8.3.4.3.6.

8.3.4.3.1 Scheduling Structure Representation

The internal structure of the Scheduler configuration tables is not exposed to software. EMP firmware is responsible for the Scheduler configuration, and provides software with an Admin Queue interface allowing alternative scheduler configuration.

Allocation and deallocation of nodes is also hidden from software and performed by EMP firmware based on software resource allocation/deallocation requests. EMP firmware manages the shared pool of the Tx-Scheduler leaf nodes as well as all other nodes.

A new added scheduling component is configured with a minimal default bandwidth allocation. Minimum bandwidth and bandwidth limit are disabled. Software can modify the default bandwidth allocations using the Admin Queue commands described in Section 8.3.4.3.6.

To create a common language between the different components in the system (drivers, firmware, and hardware) a logical representation of the Tx-Scheduler nodes is required. The objects represented in the models are identified by a device with Tx-Scheduler element identifier (TEID). The TEID of a component is returned upon creation of the component and is referenced when creating ties between components. The TEID of components created automatically can be retrieved using the "Query Default Scheduling Tree Topology" Admin Queue command. See Section 8.3.4.3.6.1 for more details.

If the scheduler runs out of resources (for example, leaf nodes), the respective allocation command fails and the requesting software is notified with an error return code.

8.3.4.3.2 Scheduler Configuration Tables

Tx-Scheduler is configured using a set of Scheduler configuration tables. Tx-Scheduler configuration tables are shared by all PCI functions, and managed by EMP firmware. Scheduler table configuration is performed by EMP firmware, and software can use Admin Queue commands described in Section 8.3.4.3.6 to alter default scheduler tables configuration.

8.3.4.3.3 Firmware Scheduler Interface

EMP firmware uses a register-based interface to access and program Tx-Scheduler configuration tables. In the normal operation mode, those registers are accessible and controlled by firmware only. In debug mode, all scheduler interface registers are mapped to the PCIe address space, and can be accessed by other CPUs or host software. No access synchronization mechanism is provided by hardware. Therefore, in debug mode, it would be up to software and firmware to coordinate their access to the Scheduler programming interface.

Scheduler provides two types of accesses to its configuration structures:

- **Immediate Access** — A basic operation allowing to read and write one entry of the scheduler configuration tables. Usually, firmware performs one table access at a time using this interface.
- **Batched Access** — A more advanced operation mode, allowing firmware to post multiple commands and have them executed as a single operation that can be executed in atomic manner when required. Scope of commands that can be used for batch operation is more limited than immediate commands, and primarily includes operations of copying an existing entry to the new location, or updating a field in the table entry.

Firmware is allowed to concurrently use both interfaces. Tx-Scheduler does not guarantee ordering between Batched and Immediate commands, and executes them in round-robin sequence along with other scheduling operations.

The table below lists all scheduler interface commands supported via Immediate or Batched interface. The majority of the commands can be posted via either one of interfaces. Any invalid command leads to a critical error and suspends a respective interface. Detected errors are reported in TSCDIFSTATUS register. The interface can be re-enabled by setting a *ICMDCLRERR* or *BCMDCLRERR* bit in the GLCD_IFCTRL register.

Submission of Immediate is performed using GLPSM_IFICMDL and GLPSM_IFICMDH CSRs. Batched commands are performed using GLPSM_IFBCMDL and GLPSM_IFBCMDH CSRs. CSR GLPSM_IFSTATUS provides the status for both interfaces. Not all the commands use the high DWord part. When the high part is required, it should be fed first. Write to low register indicates to hardware that command is ready and can be processed.

Both Immediate and Batched interface support a limited number of outstanding commands (Immediate: single command; Batched: up to 128 commands). Firmware must use the GLPSM_IFSTATUS register to validate that the interface has a free space for the new command. Overflow of immediate or batched interfaces is forbidden; any overflow command is silently dropped. EMP firmware is considered a trusted entity, and it is instructed to prevent any overflow event.

8.3.4.3.3.1 Immediate Command Interface

Firmware flow usually includes:

1. Read the GLPSM_IFSTATUS register to make sure that no Immediate command is pending. This operation can be skipped if previously posted immediate operation has been completed.
2. Write data to the GLPSM_IFDATA[n] registers (for WriteEntry operations). Order of writing data is not important.
3. Write command to the GLPSM_IFICMDH/L registers. Only few commands use the GLPSM_IFDATA[n] register. If posting a different command, firmware can skip writing to the GLPSM_IFDATA[n] register, and fill the GLPSM_IFICMDH/L register only. Content of the GLPSM_IFDATA[n] register should the be ignored.
4. Read data from the GLPSM_IFDATA[n] register for ReadEntry operations.

Though the Immediate interface is primarily designed for ReadEntry and WriteEntry operations, it can be used by firmware to post other commands.

8.3.4.3.3.2 Batched Command Interface

Batched Command interface allows firmware to post multiple commands and gain faster execution. Batch commands can be executed in atomic fashion if required. This interface allows firmware to perform the majority of Tx-Scheduler configuration tables without suspending normal Scheduler operation flow.

Batched Command interface allows firmware to post up to 128 outstanding commands. Each sequence of commands posted to Batched interface should be completed with BatchDone Control command posted to Batched interface. This command indicates that Batch sequence is completed. Commands posted to Batched interface are not executed immediately, but are queued to the 128-deep Batch FIFO. Firmware should "ring DB" by writing to the GLPSM_IFCTRL register to request execution of the posted sequence of batched commands.

Following is an example of posting Batched command sequence:

1. Read the GLPSM_IFSTATUS register to make sure that Batched interface has enough space available.
2. Write one or more commands to the GLPSM_IFBCMDH/L registers.
3. Complete the sequence by writing the BatchDone command to the GLPSM_IFBCMDL register
4. Write to the GLPSM_IFCTRL register to ring DB.

Firmware is allowed to interleave commands posted via Immediate and Batched interface. One usage model of that would be gathering information using ReadEntry Immediate commands to build a sequence of Batched commands.

Firmware is allowed to post multiple sequences of commands to Batched interface, each terminated by the BatchDone command. Each time firmware posts the BatchDone command to the Batched interface, it must ring DB. Hardware increments an internal counter with each DB ring, and decrements with each processed BatchDone command. If the counter remains positive, hardware processes the next batch after it completes a round of performing other scheduling flows.

8.3.4.3.4 LAN Queue or RDMA Queue Set Assignment

As a part of the scheduler configuration, each Tx-Scheduler Leaf node is supplied with a logical queue handle for LAN queue or RDMA Queue Set.

For basic operation (Prior adding the RDMA support), two look-up tables (LUT) are managed by EMP firmware and used for Tx-Scheduler: "Tx-Queue ID to Leaf node ID" and "Leaf node ID to Tx-Queue ID". The Tx-Queue context does not point directly to the leaf node ID. The Tx-Scheduler block activities relate to the leaf node ID. The LUTs are used in the interface between the Tx-Scheduler block and the transmit pipe. This way EMP firmware can update the internal structure and node locations internally without involving other blocks.

When RDMA is served in parallel to the LAN traffic, the RDMA Queue Set IDs are managed independently from the LAN Queue IDs. The LAN Queue at PSM interface is a number between 0-16K. The RDMA Queue Set ID is a number between 0-511. To allow parallel operation of LAN Queue and RDMA QSets, an additional table is added called "RDMA QSet ID to Leaf Node ID". This is the LUT for RDMA. For the calls from the RDMA interface, this table is used to find the Leaf Node ID. Calls from PSM to the Pipe, Leaf node ID to Tx-Queue ID include a flag in each entry that indicates whether if this call is targeted at the LAN or the RDMA interface.

The Queue or QSet ID that is propagated in the Tx-Pipe also includes the RDMA or LAN flag. The Update operation from TDPU carries this flag as well for correct update operations.

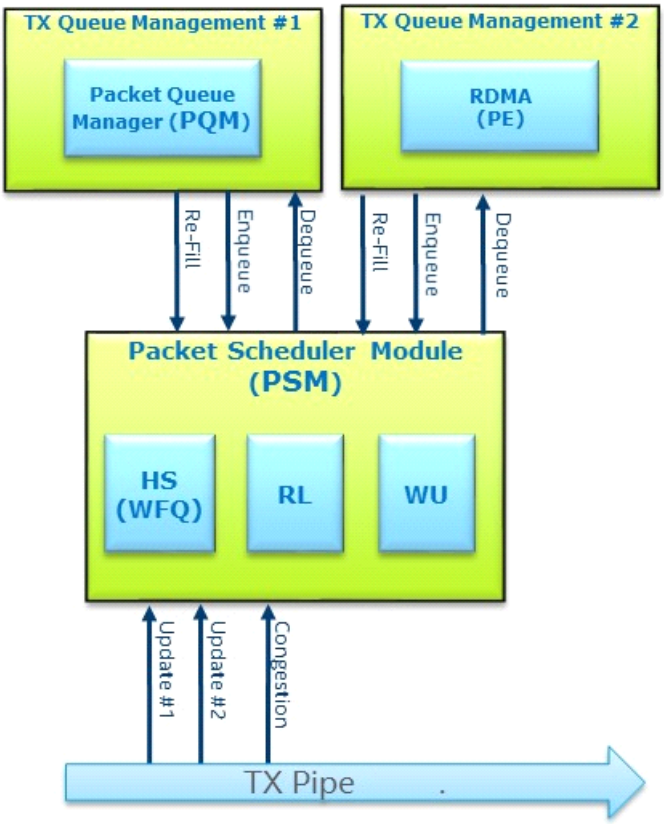


Figure 8-28. Parallel RDMA-PQM Connectivity to PSM

8.3.4.3.5 Releasing of Scheduling Elements

The software can use the AQ Command: Delete a Scheduling Component. Removal of the Scheduling Component is allowed only if the respective uplink Scheduling element has been previously removed. Software can request to remove a Leaf node only if the Queue/Queue Set associated with that Leaf node is disabled.

8.3.4.3.6 Admin Queue Commands

The internal structure of the Scheduler configuration tables is not exposed to software. This section defines Admin Queue commands used by software to program Scheduler configuration attributes.

The standard E810 Scheduler topology configuration is based on the network topology. The Scheduler configuration adds bandwidth management attributes to the configured scheduling components. Bandwidth configuration is based on the network topology deployment and capacity. The bandwidth management also reflects the QoS definition (SLA) required for the connected users and tenants.

The Admin Queue commands described in this section are intended for use by the PF driver only. The VF driver is not allowed to directly participate in the Scheduler configuration. The PF driver is considered to be a trusted software component within this PF. EMP firmware validates that Admin Queue commands do not impact configuration of components not owned by the PF driver issuing the commands. This is performed using association of the Admin Queue with particular PF and ownership information stored in the scheduling configuration tables. When the PF performs configuration on behalf of the VF, the corresponding VF must be provided with the Admin Queue command.

Table 8-24 provides a list of Admin Queue commands that allow configuration of the internal scheduler structures.

EMP firmware must avoid unrecoverable partial command execution. Some commands that configure larger amount of entities are broken into a groups of configured entities. In that case, each group must not be partially configured. All command validation and resource availability checks must be performed prior to updating Scheduler Configuration tables. Aborting a command with a partially updated Scheduler Configuration table can lead to unpredictable hardware behavior.

Notes: Bandwidth allocation setup among sibling nodes is configured in weights. As a node is configured with higher rate, it gets more bandwidth relative to its siblings. The E810 supports a 0.5% granularity of bandwidth allocation weight setup. In all bandwidth setup commands, the bandwidth allocation weight is a field of integer in the range of 1..200. Refer to Section 10.5.4.3, "LAN Transmit Queue Modes", A scheduling subtree is considered fine-grained only if all its descendant Tx-Queues are configured for "Advanced Transmit Mode" using the Quanta Queue interface. The full range of 1..200 of the bandwidth allocation is allowed only inside a fine-grained subtree. If any of the descendant Tx-Queues of the node are configured for "Legacy Mode", the node bandwidth weight range is 4..200.

It is the caller PF's responsibility to limit the range if there is a Legacy Tx-Queue in the subtree.

Table 8-24. Scheduler Configuration Admin Queue Commands

Command	Opcode	Brief Description	Section Reference
Query Default Scheduling Tree Topology	0x0400	Retrieve the scheduler topology from the EMP firmware for a given port.	8.3.4.3.6.1
Add Scheduling Elements	0x0401	Allocate a list of new scheduling elements as a child of the specified parent's TEID in the command buffer.	8.3.4.3.6.3
Configure Scheduling Elements	0x0403	Configure scheduling parameters for a list of existing scheduling elements as specified by parent's TEID.	8.3.4.3.6.4
Query Scheduling Elements Configuration	0x0404	Retrieve the bandwidth and scheduler configuration information for the list of scheduling elements as specified by their TEIDs from the EMP firmware.	8.3.4.3.6.2
Move Scheduling Element	0x0408	Move multiple siblings in a scheduling tree from one parent node to another parent node. This command works on any node in any layer. If the nodes being moved have children, the whole subtree is moved. If the old and new parent nodes belong to the same scheduling layer, the move is done without disrupting any scheduling operation.	8.3.4.3.6.5

Table 8-24. Scheduler Configuration Admin Queue Commands [continued]

Command	Opcode	Brief Description	Section Reference
Suspend Nodes	0x0409	Suspend scheduling operations for a list of scheduler elements as specified by their TEIDs. If the suspended nodes have children, the whole subtree starting as the scheduling element TEID to the leaf nodes of that element is suspended.	8.3.4.3.6.6
Resume Nodes	0x040A	Resume scheduling operations for a list of scheduler element as specified by their TEIDs; that were previously suspended using "Suspend Nodes" AQ.	8.3.4.3.6.7
Suspend Port's Tx Traffic per Traffic Class	0x040B	Suspend a port's Tx traffic for a given traffic class. The command is completed once the LAN Queues and RDMA QSets belonging to a given traffic class for the given port are suspended and drained.	8.3.4.3.6.8
Resume Port's Tx Traffic per Traffic Class	0x040C	Resume a port's Tx traffic for a given traffic class that was previously suspended.	8.3.4.3.6.9
Configure Port ETS	0x040D	Configure a port's ETS configuration. This command is typically used when software owns LLDP/DCBX processing and can also control port's ETS configuration.	8.3.4.3.6.10
Query Port ETS	0x040E	Retrieve a port's ETS configuration.	8.3.4.3.6.11
Delete Scheduling Elements	0x040F	Delete scheduling elements identified by their TEIDs. EMP Firmware checks for validity of the specified TEIDs, and if there are any children still connected to any of the TEIDs before attempting to delete the elements.	8.3.4.3.6.12
Add RL Profiles	0x0410	Add Rate Limiter profiles.	8.3.4.3.7.2
Query RL Profiles	0x0411	Retrieve a number of Rate Limiter profiles from the EMP firmware.	8.3.4.3.7.3
Remove RL Profiles	0x0415	Delete 1-4 Rate Limiter profiles.	8.3.4.3.7.4
Query Scheduling Resource Allocation	0x0412	Retrieve the schedule resources allocated by EMP firmware to the given PF.	8.3.4.3.7.1
Query Node-to-Root Topology	0x0413	Retrieve the scheduler topology from the EMP firmware for a given TEID to the Root node.	8.3.4.3.6.13
Configure L2 Node CGD	0x0414	Association of CGD's and L2 nodes.	8.3.4.3.6.14
Query L2 Node Physical ID	0x0415	Retrieve physical node ID for specified L2 nodes.	8.3.4.3.6.15

8.3.4.3.6.1 Query Default Scheduling Tree Topology (0x0400)

This command retrieves the scheduler topology from the EMP firmware for a given port.

The EMP firmware, as response to this command, returns all the scheduling elements that are visible to the software in the provided data buffer. The EMP firmware includes only the elements that were created by firmware as part of initial scheduler tree configuration creation for the given port.

This is an Indirect Admin Queue command, with additional command attributes and completion attributes provided within the data buffer. Table 8-25 describes command format and defines command-specific fields.

This command is called by PF only at init time and retrieves the default topology. Once the topology is modified (by calling a proper AQ), EMP firmware stops serving this command and returns with the error code EPERM (Operation not permitted).

Table 8-25. Query Default Scheduling Tree Topology Command and Response Fields

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2 for details.
Opcode	2-3	0x0400	
Datalen	4-5		Length of response buffer. Software provides a buffer of 4 KB. Firmware writes here the actual length of the response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16	0	The value is ignored by firmware and the port number is implicit to the calling function. Reserved. Must be set to 0.
Total Branches	17		Must be set to 0 as part of command. In Command Response this field is the total number of port-to-queue branches created by firmware (value of 1-8).
Reserved	18-23	0	Reserved. Must be set to 0.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Table 8-26 describes format of the data returned by firmware in the response buffer for each branch. EMP firmware returns all the elements starting from Root node to the Scheduling Elements it has added for a given branch.

Table 8-26. Query Default Scheduling Tree Topology Command Response Buffer per Branch Structure

Name	Byte.Bit	Value	Remarks
Reserved	0-3		
Number of Elements	4-5	2-9	Total number of elements created for given branch (TC).
Reserved	6-7		
Element 0	8-31		First element in the tree for this branch (Root node).

Table 8-26. Query Default Scheduling Tree Topology Command Response Buffer per Branch Structure [continued]

Name	Byte.Bit	Value	Remarks
. . .			
Element 9			9th element in the tree for this branch.

Table 8-27 describes format of the data returned by firmware in the response buffer for each element in the scheduling tree within a given branch.

Table 8-27. Query Default Scheduling Tree Topology Command Response Buffer per Element Structure

Name	Byte.Bit	Value	Remarks
Parent TEID	0-3		Element's parent TEID. ***Note:*** For the Element Type "Root Port" this would be set to 0xFFFFFFFF.
Node TEID	4-7		Element TEID.
Element Type	8	0 = Undefined 1 = Root Port 2 = TC 3 = SE Generic 4 = Software entry point SE 5 = Leaf 6-255 = Reserved	In some cases, Element Type can be both software entry point SE and a Leaf. In that case, the Element is marked as a software entry point SE. ***Note:*** Root Port, TC nodes are typically in firmware control.
Valid Sections	9	**Bit 9.0:** Generic section (Must be set to 1b) **Bit 9.1:** CIR BW **Bit 9.2:** EIR BW **Bit 9.3:** Shared BW **Bits 9.4-9.7:** Reserved	Multiple sections can be valid at given time. EIR BW and Shared BW profiles are mutually exclusive and hence only one of them can be set for any given element.
Generic	10	**Bit 10.0:** Scheduling Mode 0b = BPS 1b = PPS **Bits 10.1–10.3:** Priority among siblings (0-7) **Bit 10.4:** Single Priority 0b = Node 1b = WFQ **Bits 10.5-10.6:** Adjustment value (0-3) used in PSM Credit Update flow **Bit 10.7:** Reserved	
Flags	11	**Bit 11.0:** If set to 1, indicates node is suspended. **Bits 11.1-11.7:** Reserved.	
CIR BW Profile ID	12-13		
CIR BW Weight	14-15	WFQ Weight (1..200) in Precise subtree, (4..200) otherwise. (See note in Section 8.3.4.3.6)	
EIR BW Profile ID	16-17		
EIR BW Weight	18-19	WFQ Weight (1..200) in Precise subtree, (4..200) otherwise. (See note in Section 8.3.4.3.6)	
Shared RL Profile ID	20-21		
Reserved	22-23		

8.3.4.3.6.2 Query Scheduling Elements Configuration (0x0404)

This command retrieves the bandwidth and scheduler configuration information for the list of scheduling elements as specified by their TEIDs from the EMP firmware.

The software provides the list of TEIDs as part of the command buffer, whereas the EMP firmware, as response to this command, returns bandwidth information on all the scheduling elements to the software in the software-provided data buffer for the specified TEIDs. EMP firmware validates that all the TEIDs are owned by the calling PF.

This is an Indirect Admin Queue command, with additional command attributes and completion attributes provided within the data buffer. Table 8-28 describes command format and defines command-specific fields.

Table 8-28. Query Scheduling Elements Command and Response Fields

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2 for details.
Opcode	2-3	0x0404	
Datalen	4-5		Length of response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of Elements Requested	16-17		Total number of scheduling elements.
Number of Elements Returned	18-19		Command: Set to 0 by driver. Response: Firmware puts the number of elements returned.
Reserved	20-23	0	Reserved. Must be set to 0.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Table 8-29 describes the format for providing the list of scheduling elements' TEIDs by software as part of the command buffer.

Table 8-29. Query Scheduling Elements Command and Response Fields

Name	Byte.Bit	Value	Remarks
Reserved	0-3	0	
Node TEID	4-7		Element TEID of the scheduling element. Set by software for retrieving the bandwidth configuration from the firmware.
Reserved	8-23	0	

Table 8-30 describes format of the data returned by firmware in the response buffer for each element in the scheduling tree.

Table 8-30. Query Scheduling Elements Configuration Command Response Buffer per Element Structure

Name	Byte.Bit	Value	Remarks
Parent TEID	0-3		Element's parent TEID. ***Note:*** For the Element Type "Root Port" this would be set to 0xFFFFFFFF.
Node TEID	4-7		Element TEID.
Element Type	8	0 = Undefined 1 = Root Port 2 = TC 3 = SE Generic 4 = Software entry point SE 5 = Leaf 6 = SE Padded 7-255 = Reserved	***Note:*** Root Port, TC nodes are typically in firmware control.
Valid Sections	9	**Bit 9.0:** Generic section (Must be set to 1b) **Bit 9.1:** CIR BW **Bit 9.2:** EIR BW **Bit 9.3:** Shared BW **Bits 9.4-9.7:** Reserved	Multiple sections can be valid at given time.
Generic	10	**Bit 10.0:** Scheduling Mode 0b = BPS 1b = PPS **Bits 10.1–10.3:** Priority among siblings (0-7) **Bit 10.4:** Single Priority 0b = Node 1b = WFQ **Bits 10.5-10.6:** Adjustment value (0-3) used in PSM Credit Update flow **Bit 10.7:** Reserved	
Flags	11	**Bit 11.0:** If set to 1, indicates node is suspended. **Bits 11.1-11.7:** Reserved.	
CIR BW Profile ID	12-13		
CIR BW Weight	14-15	WFQ Weight (1..200) in Precise subtree, (4..200) otherwise. (See note in Section 8.3.4.3.6)	
EIR BW Profile ID	16-17		
EIR BW Weight	18-19	WFQ Weight (1..200) in Precise subtree, (4..200) otherwise. (See note in Section 8.3.4.3.6)	
Shared RL Profile ID	20-21		
Reserved	22-23		

8.3.4.3.6.3 Add Scheduling Elements (0x0401)

This command allocates a list of new scheduling elements as a child of the specified parent's TEID in the command buffer.

The EMP firmware as response to this command returns the TEID for the newly added scheduling elements for each element.

Leaf nodes must be attached to a LAN Tx-Queue or a RDMA QSet. Therefore, Leaf nodes are added only with the AQ commands field.

EMP firmware verifies that a Leaf node is not added by the "Add Scheduling Elements" command. This verification must be done prior adding any queue of the Queue Group to prevent partial execution of the command.

This is an Indirect Admin Queue command, with additional command attributes and completion attributes provided within the data buffer. Table 8-31 describes command format and defines command-specific fields.

Table 8-31. Add Scheduling Elements Command and Response Fields

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2 for details.
Opcode	2-3	0x0401	
Datalen	4-5		Length of command and response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. Response: 0 = Success. Otherwise = Not all Groups could be added successfully. Software should look at the Number of Groups Added Value and the Response Buffer for the Scheduling Elements added for those groups.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of Groups Requested	16-17		Number of Scheduling element groups to be added per Parent TEID.
Number of Groups Returned	18-19		Command: Set to 0 by driver. Response: Firmware puts the number of groups successfully added.
Reserved	20-23	0	Reserved. Must be set to 0.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Table 8-32 describes format of the command buffer per Group, and Table 8-33 describes the format of the command data set by software for each element that is part of the group. Software can create multiples of such Groups as identified by the "Number of Groups" field in the command. The EMP firmware updates the *Node TEID* field in the response buffer for each element that was added to the scheduler tree.

Table 8-32. Add Scheduling Elements Command Buffer per Group

Name	Byte.Bit	Value	Remarks
Parent TEID	0-3		
Number of Elements	4-5		
Reserved	6-7		
Element 0	8-31		Contains first Element entry (See Table 8-33).
. . .			
Element *n*			Contains *nth* Element entry.

Table 8-33. Add Scheduling Elements Command and Response Buffer per Element Structure

Name	Byte.Bit	Value	Remarks
Reserved	0-3		
Node TEID	4-7		Reserved (0): When part of command buffer, firmware sets this field to Element TEID as part of the response buffer.
Reserved	8		
Valid Sections	9	**Bit 9.0:** Generic section (Must be set to 1b) **Bit 9.1:** CIR BW **Bit 9.2:** EIR BW **Bit 9.3:** Shared BW **Bits 9.4-9.7:** Reserved	Multiple sections can be valid at given time. When only *EIR BW* is set, firmware configures EIR bandwidth only. When only *Shared BW* is set, firmware configures SRL ID and switch Node to SRL. When both *EIR BW* and *Shared BW* are set, firmware configures EIR, SRL, and switch to SRL.
Generic	10	**Bit 10.0:** Scheduling Mode 0b = BPS 1b = PPS **Bits 10.1–10.3:** Priority among siblings (0-7) **Bit 10.4:** Single Priority 0b = Node 1b = WFQ **Bits 10.5-10.6:** Adjustment value (0-3) used in PSM Credit Update flow **Bit 10.7:** Reserved	
Reserved	11		
CIR BW Profile ID	12-13		
CIR BW Weight	14-15	WFQ Weight (1..200) in Precise subtree, (4..200) otherwise. (See note in Section 8.3.4.3.6)	
EIR BW Profile ID	16-17		
EIR BW Weight	18-19	WFQ Weight (1..200) in Precise subtree, (4..200) otherwise. (See note in Section 8.3.4.3.6)	
Shared RL Profile ID	20-21		
Reserved	22-23		

8.3.4.3.6.4 Configure Scheduling Elements (0x0403)

This command configures scheduling parameters for a list of existing scheduling elements as specified by parent's TEID.

This is an Indirect Admin Queue command, with additional command attributes and completion attributes provided within the data buffer. Table 8-34 describes command format and defines command-specific fields.

Table 8-34. Configure Scheduling Elements Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0403	
Datalen	4-5		Length of response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. Response: 0 = Success. 1 = Not all Elements could be configured successfully. Software should look at the Number of Elements Configured Value and the Response Buffer for the Scheduling Elements configured.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of Elements Requested	16-17		Total number of elements of scheduling elements to be configured.
Number of Elements Configured	18-19		Command: Set to 0 by driver. Response: Firmware puts the number of configured elements in response.
Reserved	20-23	0	Reserved. Must be set to 0.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Table 8-35 describes the format of the command data set by software for each element.

Table 8-35. Configure Scheduling Elements Command and Response Buffer per Element Structure

Name	Byte.Bit	Value	Remarks
Reserved	0-3		
Node TEID	4-7		Element TEID.
Reserved	8		
Valid Sections	9	**Bit 9.0:** Generic section (Must be set to 1b) **Bit 9.1:** CIR BW **Bit 9.2:** EIR BW **Bit 9.3:** Shared BW **Bits 9.4-9.7:** Reserved	Multiple sections can be valid at given time. When only *EIR BW* is set, firmware configures EIR bandwidth only. When only *Shared BW* is set, firmware configures SRL ID and switch Node to SRL. When both *EIR BW* and *Shared BW* are set, firmware configures EIR, SRL, and switch to SRL.

Table 8-35. Configure Scheduling Elements Command and Response Buffer per Element Structure [continued]

Name	Byte.Bit	Value	Remarks
Generic	10	**Bit 10.0:** Scheduling Mode 0b = BPS 1b = PPS **Bits 10.1–10.3:** Priority among siblings (0-7) **Bit 10.4:** Single Priority 0b = Node 1b = WFQ **Bits 10.5-10.6:** Adjustment value (0-3) used in PSM Credit Update flow **Bit 10.7:** Reserved	
Reserved	11		
CIR BW Profile ID	12-13		
CIR BW Weight	14-15	WFQ Weight (1..200) in Precise subtree, (4..200) otherwise. (See note in Section 8.3.4.3.6)	
EIR BW Profile ID	16-17		
EIR BW Weight	18-19	WFQ Weight (1..200) in Precise subtree, (4..200) otherwise. (See note in Section 8.3.4.3.6)	
Shared RL Profile ID	20-21		
Reserved	22-23		

8.3.4.3.6.5 Move Scheduling Elements (0x0408)

This command allows software to move multiple siblings in a scheduling tree from one parent node to another parent node. This command works on any node in any layer, except the *root* and *leaf* layers. If the nodes being moved have children, the whole subtree is moved. If the old and new parent nodes belong to the same scheduling layer, the move is done without disrupting any scheduling operation.

This is an Indirect Admin Queue command, with additional command attributes and completion attributes provided within the data buffer. Table 8-36 describes command format and defines command-specific fields.

Table 8-36. Configure Scheduling Elements Command and Response Buffer per Element Structure

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2 for details.
Opcode	2-3	0x0408	
Datalen	4-5		Length of response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of Sibling Groups	16-17		Set by the driver. Unchanged by the firmware.
Number of Actually Moved Sibling Groups	18-19		Zeroed by the driver. Set by the firmware.
Reserved	20-23	0	Reserved. Must be set to 0.

Table 8-36. Configure Scheduling Elements Command and Response Buffer per Element Structure [continued]

Name	Byte.Bit	Value	Remarks
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Table 8-37 describes the per sibling group structure that provided by software.

Table 8-37. Move Scheduling Elements Buffer per Sibling Group

Name	Byte.Bit	Value	Remarks
Existing Parent TEID	0-3[1]		Current TEID of the parent of the siblings being moved in this group.
New Parent TEID	4-7		New Parent TEID where the siblings are moved to as children.
Number of Siblings to be Moved	8-9	0	
Reserved	10-11		

1. The existing parent must be a direct parent of the moved nodes in the logical space (hidden layers remain hidden for that command). EMP firmware enforces this and returns error if required.

Software provides a list of Scheduling Element TEIDs for each sibling as part of the command buffer.

Table 8-38 describes the per-sibling TEID provided by software.

Table 8-38. Move Scheduling Elements Buffer per Sibling in the Command Buffer

Name	Byte.Bit	Value	Remarks
Element TEID	0-3		

Table 8-39 describes an example of the command buffer (***n*** Siblings need to be moved by software).

Table 8-39. Move Scheduling Elements Buffer per Sibling in the Command Buffer

Bytes	Name
0-11	Sibling Group #0
12-15	Element #0
	. . .
	Element #n-1
	. . .
	Sibling Group #m-1
	Element #0
	. . .
	Element #n-1

8.3.4.3.6.6 Suspend Nodes (0x0409)

This command allows software to suspend scheduling operations for a list of scheduler elements as specified by their TEIDs.

If the suspended nodes have children, the whole subtree starting at the scheduling element TEID to the leaf nodes of that element are not scheduled until the suspended parent is resumed.

This is an Indirect Admin Queue command, with additional command attributes and completion attributes provided within the data buffer. Table 8-40 describes command format and defines command-specific fields.

Table 8-40. Suspend Node Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0409	
Datalen	4-5		Length of response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of Elements	16-17		Number of Elements to be suspended.
Number of Actually Suspended Elements	18-19		Zeroed by the driver. Written by the EMP firmware.
Reserved	20-23	0	Reserved. Must be set to 0.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Table 8-41. Suspend Nodes Buffer per Element in the Command Buffer

Name	Byte.Bit	Value	Remarks
Element TEID	0-3		

8.3.4.3.6.7 Resume Nodes (0x040A)

This command allows software to resume scheduling operations for a list of scheduler elements as specified by their TEIDs; that were previously suspended using "Suspend Nodes" AQ.

This is an Indirect Admin Queue command, with additional command attributes and completion attributes provided within the data buffer. Table 8-42 describes command format and defines command-specific fields.

Table 8-42. Resume Node Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x040A	
Datalen	4-5		Length of response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of Elements	16-17		Number of Elements to be resumed.
Number of Actually Resumed Elements	18-19		Zeroed by the driver. Written by the EMP firmware.
Reserved	20-23	0	Reserved. Must be set to 0.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Table 8-43. Resume Nodes Buffer per Element in the Command Buffer

Name	Byte.Bit	Value	Remarks
Element TEID	0-3		

8.3.4.3.6.8 Suspend Port's Tx Traffic per Traffic Class (0x040B)

This command allows software to suspend a port's Tx traffic for a given traffic class. The command is completed once the LAN Queues and RDMA QSets belonging to given traffic class for the given port are suspended and drained.

This helps software to modify port's traffic class configuration whenever there are changes required due to DCB/DCBX settings when software owns LLDP/DCBX processing.

This is a Direct Admin Queue command with additional command attributes, and completion attributes are provided within the data buffer. Table 8-44 describes command format and defines command-specific fields.

Table 8-44. Suspend Port's Tx Traffic per Traffic Class Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x040B	
Datalen	4-5		Length of response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
TC Bitmap	16		Bitmap of Traffic Classes to be suspended.
Reserved	17-23	0	Reserved. Must be set to 0.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

8.3.4.3.6.9 Resume Port's Tx Traffic per Traffic Class (0x040C)

This command allows software to resume a port's Tx traffic for a given traffic class that was previously suspended

This command is used by software to resume port's Tx traffic for a TC after port's traffic class configuration was changed due to DCB/DCBX.

This is a Direct Admin Queue command with additional command attributes, and completion attributes are provided within the data buffer. Table 8-45 describes command format and defines command-specific fields.

Table 8-45. Resume Port's Tx Traffic per Traffic Class Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.1.1 for details.
Opcode	2-3	0x040C	
Datalen	4-5		Length of response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
TC Bitmap	16		Bitmap of Traffic Classes to be resumed.
Reserved	17-23	0	Reserved. Must be set to 0.

Table 8-45. Resume Port's Tx Traffic per Traffic Class Command [continued]

Name	Byte.Bit	Value	Remarks
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

8.3.4.3.6.10 Configure Port ETS (0x040D)

This command allows software to configure a port's ETS configuration. This command is typically used when software owns LLDP/DCBX processing and can also control port's ETS configuration. When LLDP/DCBX processing is owned by the EMP firmware, this command returns an EPERM error.

This is an Indirect Admin Queue command, with additional command attributes and completion attributes provided within the data buffer. Table 8-46 describes command format and defines command-specific fields.

Table 8-46. Configure Port ETS Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x040D	
Datalen	4-5		Length of response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Port TEID	16-19		Root Port TEID.
Reserved	20-23	0	Reserved. Must be set to 0.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Table 8-47 describes format of the command response buffer. In the response, TEID is only valid for TCs that are configured, else it is set to INVALID_TEID (0xFFFFFFFF).

Table 8-47. Configure PORT ETS Command and Response Buffer

Name	Byte.Bit	Value	Remarks
Enabled TC Bitmap	0	TC0-TC7	Bitmap of enabled TCs at the port. Set by the driver. Untouched by the firmware.
Reserved	1-3		
Per Priority TC	4-7	**Bits 4.0-4.3**: UP0 TC **Bits 4.4-4.7:** UP1 TC **Bits 5.0-5.3:** UP2 TC **Bits 5.4-5.7:** UP3 TC **Bits 6.0-6.3:** UP4 TC **Bits 6.4-6.7:** UP5 TC **Bits 7.0-7.3:** UP6 TC **Bits 7.4-7.7:** UP7 TC	Currently, this is a placeholder field and is not processed. Set by the driver. Untouched by the firmware.

Table 8-47. Configure PORT ETS Command and Response Buffer [continued]

Name	Byte.Bit	Value	Remarks
TC Bandwidth Share	8-15	**Byte 8:** TC0 bandwidth **Byte 9:** TC1 bandwidth **Byte 10:** TC2 bandwidth **Byte 11:** TC3 bandwidth **Byte 12:** TC4 bandwidth **Byte 13:** TC5 bandwidth **Byte 14:** TC6 bandwidth **Byte 15:** TC7 bandwidth	Each byte represents TC Bandwidth Ratio (%) relative to other TCs on the specified port. Set by the driver. Untouched by the firmware.
Port BW Limit Profile	16-19	**Byte 16-17:** EIR Profile ID **Byte 18-19:** Reserved	The Profile ID for Port (Root Node) is fixed, and software must configure the Root Node's RL Profile using "Configure RL Profile" to modify this. Set by the driver. Untouched by the firmware.
Reserved	20-23		
TC Node Priority	24-27	**Bits 24.0-24.3:** TC0 Node Priority **Bits 24.4-24.7:** TC1 Node Priority **Bits 25.0-25.3:** TC2 Node Priority **Bits 25.4-25.7:** TC3 Node Priority **Bits 26.0-26.3:** TC4 Node Priority **Bits 26.4-26.7:** TC5 Node Priority **Bits 27.0-27.3:** TC6 Node Priority **Bits 27.4-27.7:** TC7 Node Priority	Set by the driver. Untouched by the firmware. Valid Values in each TC: 0-7
Reserved	28-31		
TC0 Node TEID	32-35		Zeroed by the driver. Written by the firmware.
TC1 Node TEID	36-39		
TC2 Node TEID	40-43		
TC3 Node TEID	44-47		
TC4 Node TEID	48-51		
TC5 Node TEID	52-55		
TC6 Node TEID	56-59		
TC7 Node TEID	60-63		

8.3.4.3.6.11 Query Port ETS (0x040E)

This command allows software to retrieve a port's ETS configuration.

This is an Indirect Admin Queue command, with additional command attributes and completion attributes provided within the data buffer. Table 8-48 describes command format and defines command-specific fields.

Table 8-48. Query Port ETS Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x040E	
Datalen	4-5		Length of response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.

Table 8-48. Query Port ETS Command [continued]

Name	Byte.Bit	Value	Remarks
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Port TEID	16-19		Root Port TEID.
Reserved	20-23	0	Reserved. Must be set to 0.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Table 8-49 describes format of the command buffer to be returned by firmware to software as a response.

Table 8-49. Query PORT ETS Response Buffer

Name	Byte.Bit	Value	Remarks
Enabled TC Bitmap	0	TC0-TC7	Bitmap of enabled TCs at the port.
Reserved	1-3		0
Per Priority TC	4-7	**Bits 4.0-4.3**: UP0 TC **Bits 4.4-4.7:** UP1 TC **Bits 5.0-5.3:** UP2 TC **Bits 5.4-5.7:** UP3 TC **Bits 6.0-6.3:** UP4 TC **Bits 6.4-6.7:** UP5 TC **Bits 7.0-7.3:** UP6 TC **Bits 7.4-7.7:** UP7 TC	Currently, this is a placeholder field and is not processed.
TC Bandwidth Share	8-15	**Byte 8:** TC0 bandwidth **Byte 9:** TC1 bandwidth **Byte 10:** TC2 bandwidth **Byte 11:** TC3 bandwidth **Byte 12:** TC4 bandwidth **Byte 13:** TC5 bandwidth **Byte 14:** TC6 bandwidth **Byte 15:** TC7 bandwidth	Each byte represents TC Bandwidth Ratio (%) relative to other TCs on the specified port.
Port BW Limit Profile	16-19	**Byte 16-17:** EIR Profile ID **Byte 18-19:** Reserved	The Profile ID for Port (Root Node) is fixed and, software must configure the Root Node's RL Profile using "Configure RL Profile" to modify this.
Reserved	20-23		
TC Node Priority	24-27	**Bits 24.0-24.3:** TC0 Node Priority **Bits 24.4-24.7:** TC1 Node Priority **Bits 25.0-25.3:** TC2 Node Priority **Bits 25.4-25.7:** TC3 Node Priority **Bits 26.0-26.3:** TC4 Node Priority **Bits 26.4-26.7:** TC5 Node Priority **Bits 27.0-27.3:** TC6 Node Priority **Bits 27.4-27.7:** TC7 Node Priority	
Reserved	28-31		
TC0 Node TEID	32-35		
TC1 Node TEID	36-39		
TC2 Node TEID	40-43		
TC3 Node TEID	44-47		

Table 8-49. Query PORT ETS Response Buffer [continued]

Name	Byte.Bit	Value	Remarks
TC4 Node TEID	48-51		
TC5 Node TEID	52-55		
TC6 Node TEID	56-59		
TC7 Node TEID	60-63		

8.3.4.3.6.12 Delete Scheduling Elements (0x040F)

This command allows deletion of a scheduling elements identified by their TEIDs. EMP firmware checks for validity of the specified TEIDs, and if there are any children still connected to any of the TEIDs before attempting to delete the elements (EPERM error code is return if software attempts to remove a node that is associated with children). If the TEID for a specified element is a Leaf Node with a queue associated with it, EMP firmware should return an error (EPERM).

ENOENT error code is returned if software attempts to delete an non-existent TEID.

Leaf nodes must be attached to a LAN Tx-Queue or a RDMA QSet. Therefore, Leaf nodes are deleted only with the AQ command field.

EMP firmware verifies that a Leaf node is not added by the "Delete Scheduling Elements" command. This verification must be done prior deleting any queue of the Queue Group to prevent partial execution of the command.

This is an Indirect Admin Queue command, with additional command attributes and completion attributes provided within the data buffer. Table 8-50 describes command format and defines command-specific fields.

Table 8-50. Delete Scheduling Elements Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x040F	
Datalen	4-5		Length of response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. Response: 0 = Success. Otherwise = Not all Groups could be deleted successfully. Software should look at the Number of Deleted Groups Value and the Response Buffer for the Group of Scheduling Elements deleted.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of Groups Requested	16-17		Number of Group of Elements requested to be deleted.
Number of Groups Deleted	18-19		Command: Set to 0 by driver. Response: Firmware puts the number of successfully deleted groups.
Reserved	20-23	0	Reserved. Must be set to 0.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Table 8-51 lists the content of the data buffer provided by software to delete the scheduling elements.

Table 8-51. Delete Scheduling Elements Command Buffer per Group

Name	Byte.Bit	Value	Remarks
Group 0 Parent TEID	0-3		
Number of Elements	4-5		
Reserved	6-7		
Element 0 TEID	8-11		Element TEID.
. . .			
Element *n* TEID			
Group 1 Parent TEID			
Number of Elements			
Element 0 TEID			
. . .			
Element *n* TEID			
. . .			
Group *n* Parent TEID			
Number of Elements			
Element 0 TEID			
. . .			
Element *n* TEID			

8.3.4.3.6.13 Query Node-to-Root Topology (0x0413)

This command retrieves the scheduler topology from the EMP firmware for a given TEID to the Root Node.

The EMP firmware, as response to this command, returns all the scheduling elements that are visible to the software in the provided data buffer, including any padded layers it might have added for the given hierarchy.

This is an Indirect Admin Queue command, with additional command attributes and completion attributes provided within the data buffer. Table 8-52 describes command format and defines command-specific fields.

Table 8-52. Query Node-to-Root Topology Command and Response Fields

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2 for details.
Opcode	2-3	0x0413	
Datalen	4-5		Length of response buffer.
Return Value	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.

Table 8-52. Query Node-to-Root Topology Command and Response Fields [continued]

Name	Byte.Bit	Value	Remarks
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
TEID	16-19		Element TEID.
Number of Nodes	20-23	0	Zeroed by driver. EMP firmware returns the number of elements in the branch.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Table 8-53 describes format of the data returned by firmware in the response buffer for each element in the hierarchy of scheduling tree for the given TEID.

Table 8-53. Query Node-to-Root Topology Response Buffer per Element Structure

Name	Byte.Bit	Value	Remarks
Parent TEID	0-3		Element's parent TEID. ***Note:*** For the Element Type "Root Port" this would be set to 0xFFFFFFFF.
Node TEID	4-7		Element TEID.
Element Type	8	0 = Undefined 1 = Root Port 2 = TC 3 = SE Generic 4 = Software entry point SE 5 = Leaf 6 = SE Padded 7-255 = Reserved	Root Port, TC nodes are typically in firmware control.
Valid Sections	9	**Bit 9.0:** Generic section (Must be set to 1b) **Bit 9.1:** CIR BW **Bit 9.2:** EIR BW **Bit 9.3:** Shared BW **Bits 9.4-9.7:** Reserved	Multiple sections can be valid at given time. *EIR BW* and *Shared BW* profiles are mutually exclusive and hence only one of them can be set for any given element.
Generic	10	**Bit 10.0:** Scheduling Mode 0b = BPS 1b = PPS **Bits 10.1–10.3:** Priority among siblings (0-7) **Bit 10.4:** Single Priority 0b = Node 1b = WFQ **Bits 10.5-10.6:** Adjustment value (0-3) used in PSM Credit Update flow **Bit 10.7:** Reserved	
Flags	11	**Bit 11.0:** If set to 1, indicates node is suspended. **Bits 11.1-11.7:** Reserved.	
CIR BW Profile ID	12-13		
CIR BW Weight	14-15	WFQ Weight (1..200) in Precise subtree, (4..200) otherwise. (See note in Section 8.3.4.3.6)	
EIR BW Profile ID	16-17		When the valid section has *Shared BW* field set this field points to Share RL Profile ID. When the valid section has *EIR BW* field set this field points to EIR RL Profile ID.

Table 8-53. Query Node-to-Root Topology Response Buffer per Element Structure

Name	Byte.Bit	Value	Remarks
EIR BW Weight	18-19	WFQ Weight (1..200) in Precise subtree, (4..200) otherwise. (See note in Section 8.3.4.3.6)	
Shared RL Profile ID	20-21		
Reserved	22-23		

8.3.4.3.6.14 Configure L2 Node CGD (0x0414)

This command allows configuring a congestion domain for given L2 node TEIDs in the scheduler topology.

This is an Indirect Admin Queue command, with additional command attributes and completion attributes provided within the data buffer. Table 8-54 describes command format and defines command-specific fields.

Table 8-54. Configure L2 Node CGD Command and Response Fields

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2 for details.
Opcode	2-3	0x0415	
Datalen	4-5		Length of response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of L2 Nodes	16-17		Number of L2 Nodes (1-64).
Reserved	18-23	0	Reserved. Must be set to 0.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Table 8-55 describes format of the data sent by software as part of the command buffer for each L2 scheduling element

Table 8-55. Configure L2 Node CGD Command Buffer per Element Structure

Name	Byte.Bit	Value	Remarks
Node TEID	0-3		L2 Node Element TEID
CGD	4	CGD value (0-31)	Root Port, TC nodes are typically in firmware control.
Reserved	5-7		

8.3.4.3.6.14.1 Default CGD Setup per Usage Model

Four or less ports setup:

Eight CGDs and eight L2 nodes are statically allocated in order per each port. In each port's space, the CGD's and the L2 are associated in one-to-one mapping.

Eight ports setup:

Four CGD's are statically allocated in order per each port (0-3 for port #0...). Eight L2 nodes are statically allocated in order per each port (0-7 for port #0...). In each port's space, one L2 node is associated with one CGD's. Four L2 nodes are free for PF's configuration.

8.3.4.3.6.15 Query L2 Node Physical ID (0x0415)

This command allows querying a physical node ID for given L2 node TEIDs in the Tx-Scheduler topology.

This is an Indirect Admin Queue command with additional command attributes and completion attributes are provided within the data buffer. Table 8-56 describes command format and defines command specific fields.

Table 8-56. Query L2 Node Physical ID Command and Response Fields

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2 for details.
Opcode	2-3	0x0414	
Datalen	4-5		Length of response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. EINVAL = If software sets the Number of L2 nodes to 0. EINVAL = If any of the specified L2 Node TEIDs do not exist or is invalid value (0xFFFFFFFF).
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of L2 Nodes	16-17		Number of L2 Nodes (1-64).
Reserved	18-23	0	Reserved. Must be set to 0.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Table 8-57 describes format of the data sent by software as part of the command buffer for each L2 scheduling element.

Table 8-57. Query L2 Node Physical ID Command and Response Buffer per Element Structure

Name	Byte.Bit	Value	Remarks
Node TEID	0-3		L2 Node Element TEID
Physical Node ID	4-7		Command: This is a reserved field when sent as part of command and should be set to 0. Response: Firmware will fill in the L2 node's Physical Node ID in response in this location.

8.3.4.3.7 Tx-Scheduler Resource Allocation and Management

All Tx-Scheduler entities are shared and can be used by any PF. The EMP firmware manages two types of Tx-Scheduler shared resources:

- **Generic Tx-Scheduler topology structure:**
  - This structure describes the generic scheme of Tx-Scheduler configuration. It includes the following:
    - The number of supported layers in the Tx-Scheduler topology.
    - The number of CGDs available for the entire device and for the calling PF.
    - The number of available RDMA QSets.
  - In addition, this type includes also a definition per logical layer of Max number of siblings available in each sibling group and the definition of the chunk size. Chunk size defines the granularity of resource allocation in this layer.
  - This structure is built at reset, based on NVM configuration.
  - This structure is static and is not changed during runtime.
- **Dedicated usage resources:**
  - This type of resources includes, Nodes, Profiles, and Shared rate limiters.
  - It is managed separately per each logical layer.
  - Each PF is allocated with a basic set of guaranteed resources, in addition to a shared pool of resources.
  - When a PF allocates a resource, EMP firmware allocates it from its guaranteed pool. If the guaranteed pool is exhausted and the shared pool is not, the resource is allocated from the shared pool.
  - On resource deallocation, the resource is first released from the shared pool. If the PF shared pool usage == 0, a guaranteed pool is released.
  - Once a resource of type is allocated for a PF, it is uniquely used by this PF until it is released by its owner.

Notes: An exception for the above: Rate Limit Profile #0 in both CIR and EIR is pre-allocated by the device. CIR profile #0 represents a “no committed rate” profile. EIR profile #0 represents a “no RL” profile. Those profiles are available for shared usage of all the PF's. Those Profiles are owned by EMP firmware and must not be neither modified nor deleted by any one of the PF's.

In the hardware, setting a profile which all its bits are set to zero, configuring it to “No RL” in the EIR or to “No committed rate” in the CIR.

Beside those two RL profile #0, no other RL profile can be configured same way. Any attempt of a PF to configure a profile where all its bits are set to zero, is rejected by the EMP firmware.

8.3.4.3.7.1 Query Scheduler Resource Allocation (0x0412)

This command retrieves the schedule resources allocated by EMP firmware to the given PF.

This is an Indirect Admin Queue command, with additional command attributes and completion attributes provided within the data buffer. Table 8-58 describes command format and defines command-specific fields.

Table 8-58. Query Scheduler Resource Allocation Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0412	Command opcode.
Datalen	4-5	(2-2048)	Length of response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-23		Reserved.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Table 8-59 lists the content of the data buffer received by software as a response to Query Scheduler Resource Allocation command.

Table 8-59. Query Scheduler Resource Allocation Command Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0412	Command opcode.
Datalen	4-5	16 + 32x9	Length of response buffer. Should be Number of resource returned x resource entry size.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-23		0
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Table 8-60 lists the content of the data buffer received by software as a response to Query Scheduler Resource Allocation command. Table 8-61 and Table 8-62 lists the content of data specific to different resource types.

Table 8-60. Query Scheduler Resource Allocation Response Buffer

Offset (Bytes)	Description
0-31	Generic Scheduler properties.
32-63	Logical Layer 1 properties.
. . .	
(32 x n) - (32 x (n + 1) - 1)	Logical Layer n properties (maximum 9 layers).

Table 8-61. Data Structure for Generic Scheduler Properties

Offset (Bytes)	Description
0-1	Total number of available physical scheduling layers.
2-3	Total number of available logical scheduling layers (software visible).
4	A bitmap indicating for layers 2-9, the layers that are flattened if flattening is enabled. Bit 0 points to L2 and Bit 7 points to L9
5	Total Number of CGDs supported by the device.
6	Total Number of CGDs available for this PF.
7	Reserved.
8-9	Total number of RDMA QSets in the device.
10-31	Reserved (for future use).

Table 8-62. Data Structure for Each Logical Layer

Offset (Bytes)	Description
0	Logical Layer number.
1	Chunk Size: Amount of nodes which allocated as a chunk whenever a PF gains nodes.
2-3	Total number of nodes that can be configured at this layer for the whole device.
4-9	Reserved
10-11	Max sibling group size at this layer.
12-13	Shared number of CIR RL Profiles available at this layer for the PF, including its dedicated and the available shared ones.
14-15	Shared number of EIR RL Profiles available at this layer for the PF, including its dedicated and the available shared ones.
16-17	Shared number of SRL Profiles available at this layer for the PF, including its dedicated and the available shared ones.
18-31	Reserved.

8.3.4.3.7.2 Add RL Profiles (0x0410)

This Command adds a number of Rate Limiter Profiles.

This is an Indirect Admin Queue command, with additional command attributes and completion attributes provided within the data buffer. Table 8-63 describes command format and defines command-specific fields.

Table 8-63. Add RL Profiles Command and Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2 for details.
Opcode	2-3	0x0410	
Datalen	4-5		Length of response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number or Profiles	16-17		Number of RL profiles to be retrieved for given PF.
Number of Actually Added Profiles	18-19	Zeroed by software. Written by EMP firmware.	Number of added profiles by EMP firmware.
Reserved	20-23	0	Reserved. Must be set to 0.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Table 8-64 lists the content of the data buffer provided by software to configure RL profiles for each profile.

Table 8-64. Add RL Profiles Data in Command Buffer per Each RL Profile

Name	Byte.Bit	Value	Remarks
Tx-Scheduler Layer	0	Valid values = 1-9	Logical Layer number visible to software (Max = 9). ***Note:*** In case of RLs aimed to Queues, the layer is the max logical layer independently of where the queue was attached.
Flags	1	**Bits 1.0-1.1:** Profile Type 00b = CIR 01b = EIR 10b = Shared 11b = Reserved **Bits 1.2-1.7:** Reserved (0)	Type of RL profile to be added.
Profile ID	2-3		Command: Reserved as part of command (0). Response: The profile index value returned by firmware.
Max Burst Size	4-5	Max valid value is 0xFFF (12 bits in hardware).	Max burst size.
RL Multiply	6-7	Max valid value is 0x7FF (11 bits in hardware).	Reserved when adding shared RL.

Table 8-64. Add RL Profiles Data in Command Buffer per Each RL Profile [continued]

Name	Byte.Bit	Value	Remarks
Wake-Up Calculation	8-9	When 9.7 is set to 0b: Bits 8.0-9.0 = Sub-integer granularity of number of cycles (1/128…127/128). Bits 9.1-9.6 = Number of cycles. When 9.7 is set to 1b: Bits 8.0-9.6 = Number of cycles.	Reserved when adding shared RL.
RL Encode	10-11	Max valid value is 0x3F (6 bits in hardware).	Reserved when adding shared RL.

Note: As noted above, software is allowed to add a profile where all its bits are set to zero. Any attempt to do this is rejected by EMP firmware with the error code EPERM.

8.3.4.3.7.3 Query RL Profiles (0x0411)

This command retrieves a number of Rate Limiter Profiles from the EMP firmware.

This is an Indirect Admin Queue command, with additional command attributes and completion attributes provided within the data buffer. Table 8-65 describes command format and defines command-specific fields.

Table 8-65. Query RL Profiles Command and Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0411	
Datalen	4-5		Length of response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number or Profiles	16-17		Number of RL profiles to be retrieved for given PF.
Number of Retrieved Profiles	18-23		Number of actually retrieved profiles. If this number is not equal to the requested number, an error is reported as well.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Table 8-66 lists the content of the data buffer set by software as part of the Query RL Profile command for each requested profile.

Table 8-66. Query RL Profile Data in Command Buffer for Each Profile

Name	Byte.Bit	Value	Remarks
Tx-Scheduler Layer	0	Valid values = 1-9	Logical Layer number visible to software (Max = 9).
Flags	1	**Bits 1.0-1.1:** Profile Type 00b = CIR 01b = EIR 10b = Shared 11b = Reserved **Bits 1.2-1.7:** Reserved (0)	Type of RL profile to be retrieved.
Profile ID	2-3		The profile index value.
Reserved	4-11		

Table 8-67 lists the content of the data buffer received by software as a response to Query RL Profile command.

Table 8-67. Query RL Profile Data in Response Buffer for Each Profile

Name	Byte.Bit	Value	Remarks
Tx-Scheduler Layer	0	Valid values = 1-9	Logical Layer number visible to software (Max = 9).
Flags	1	**Bits 1.0-1.1:** Profile Type 00b = CIR 01b = EIR 10b = Shared 11b = Reserved **Bits 1.2-1.6:** Reserved (0) **Bit 1.7:** Profile Allocation 0b = Profile is allocated. 1b = Profile is not allocated by any PF.	Type of RL profile to be retrieved.
Profile ID	2-3		The profile index value.
Max Burst Size	4-5	0 - 0xFFF	Max burst size.
RL Multiply	6-7	0- 0x7FF	Reserved when retrieving shared RL.
Wake-Up Calculation	8-9	When 9.7 is set to 0b: Bits 8.0-9.0 = Sub-integer granularity of number of cycles (1/128…127/128). Bits 9.1-9.6 = Number of cycles. When 9.7 is set to 1b: Bits 8.0-9.6 = Number of cycles.	Reserved when retrieving shared RL.
RL Encode	10-11	0 - 0x3F	Reserved when retrieving shared RL.

8.3.4.3.7.4 Remove RL Profiles (0x0415)

This command allows PF software to delete a number of Rate Limiter Profiles.

It is the PF's responsibility to verify that each deleted profile is not associated with any node prior to removing it. It is the PF's responsibility to limit itself for removing profiles which it owns only.

PF must never remove RL profile #0 of CIR or EIR. Attempting to remove a RL profile #0, is rejected by EMP firmware with the Error code EPERM.

This is an Indirect Admin Queue command. Table 8-68 describes command format and defines command-specific fields.

Table 8-68. Remove RL Profile Command and Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2 for details.
Opcode	2-3	0x415	
Datalen	4-5	0	Length of response buffer.
Return Value	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of Profiles to Remove	16-17		
Number of Actually Removed RL Profiles	18-19		Zeroed by the driver. Written by the EMP firmware.
Reserved	20-23		
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Table 8-69 lists the content of the data buffer set by software as part of the Remove RL Profile command for each requested profile

Table 8-69. Remove RL Profile Data in Command Buffer for Each Profile

Name	Byte.Bit	Value	Remarks
Tx-Scheduler Layer	0	Valid values = 1-9	Logical Layer number visible to software (Max = 9).
Flags	1	**Bits 1.0-1.1:** Profile Type 00b = CIR 01b = EIR 10b = Shared 11b = Reserved **Bits 1.2-1.7:** Reserved (0)	Type of RL profile to be retrieved.
Profile ID	2-3		The profile index value. Valid values according to layer. If software provides out-of-range value, EMP firmware responds with ERANGE. Same response if ID == 0.
Reserved	4-11		

The generic structure of the Admin Queue command is defined in Section 9.5.5. All the Transmit Scheduler Admin Queue commands described in the following sections are derived from the generic Admin Queue commands, using a generic Admin Queue command structure as a baseline.

Sections describing Transmit Scheduler Admin Queue commands relate to the Admin Queue command fields that are specific to the particular command. For a description of the common Admin Queue command fields, see Section 9.5.5.

The Transmit Scheduler uses various types of Admin Queue commands. Some of them are Direct Admin Queue commands (all command information is provided within the body of the command) and some are Indirect (additional data is provided within the buffer referred to by the Admin Queue command). Some commands report completion within the Admin Queue command body and others carry completion information in the buffer provided by the original command. Each Transmit Scheduler Admin Queue command indicates its type in the command description. For a detailed description of all the Admin Queue command types and their differences, see Section 9.5.5.

8.3.4.3.8 Common Processing and Error Handling

Transmit Scheduler Admin Queue commands use a generic error reporting structure described in Section 9.5.9. Table 8-70 lists errors specific to the Transmit Scheduler Admin Queue commands and reported in Admin Queue Completions, providing their cause and a reported error code.

Table 8-70. Transmit Scheduler Admin Queue Completion Errors

Error Name	Error Code	Description
Invalid Handle	ENOENT	Upon allocation of the Transmit Scheduler resources, software is provided with various handles, such as TC Handle, UP Handle, and QS Handle. Those handles must be used by software for future reference to the allocated resources. This error indicates that the handle provided by the software is not valid.
Invalid TEID	ENOENT	TEID provided within Admin Queue command is not valid.
Parameter out of range	ERANGE	Provided argument does not match definition (for example, not within allowed range).
Resource allocation failure	ENOSPC	Failed to allocate Transmit Scheduler resource.
Operation not permitted	EPERM	Depending on the scheduler configuration scheme, certain Admin Queue commands should not be used by the software (for example, ETS-based configuration software should not attempt configure bandwidth allocation to a VSI or Scheduling Component). This error indicates that the software attempted to execute an Admin Queue command that is not valid in the current context.

Chapter 9 Device Services

9.1 Interrupts

9.1.1 Interrupt Signaling

The E810 supports the following interrupt signaling according to per-PF setting options:

- Legacy INTA/INTB/INTC/INTD interrupt message on the PCIe is supported for the PFs. The E810 exposes legacy interrupt support in the *Interrupt Pin* field in the PCI configuration space of the PF. The *Interrupt Pin* parameter is loaded from the NVM (to the PFPCI_CNF register), defining the interrupt pin (A,B,C,D or none) per PF. It is the NVM programmer's responsibility to follow the PCI rules for allocating orderly interrupt pins for the PCI functions. The PFPCI_CNF register can be modified in the NVM using the adaptive NVM admin command. The legacy interrupt is triggered by the interrupt vector zero of the PF.
- MSI is exposed in the MSI capability structure in the PCI configuration space of each PF. The MSI interrupt is triggered by the interrupt vector zero of the PF. The MSI capability is enabled per PF by the *MSI_EN* bit in the PFPCI_CNF register (loaded from the NVM). The PFPCI_CNF register can be modified in the NVM using the adaptive NVM admin command.
- MSI-X enables multiple interrupts for the PFs as well as the VFs. MSI-X is exposed in the MSI-X capability structure in the PCI configuration space of all PFs and VFs. The number of supported MSI-X vectors is defined by the *Table Size* parameters in the MSI-X capability structure in the PCI configuration space of the PFs and the VFs. The *Table Size* parameters are loaded from the NVM for each PF and for each of its VFs following FLR. The *Table Size* parameters setting in the NVM (according to the values loaded to the PFINT_ALLOC.*FIRST*/*LAST* register) must obey the following limitations:
  - The total number of MSI-X vectors in the device is 2048. The sum of the *Table Size* parameters for all functions (PFs and VFs) must not exceed it.
  - The maximum number of supported MSI-X vectors for the PFs is 2048.
  - The maximum number of supported MSI-X vectors for the VFs is 2048.
  - The interrupt vectors of the functions are mapped to the internal physical space of the device.
  - The settings of *FIRST* and *LAST* fields in the PFINT_ALLOC registers can be modified in the NVM using the adaptive NVM admin command.
  - The mapping of all VFs of a PF must be contained within the space of the PF (shown in the example in Figure 9-1). The GLINT_VECT2FUNC registers provide the inverse mapping from the internal physical space to the functions. At the PF driver initialization phase, the software should set the registers of the PF interrupts according to the PFINT_ALLOC and VPINT_ALLOC settings.
  - The mappings for VFs ownership are loaded by software to the VPINT_ALLOC and GLINT_VECT2FUNC registers when allocating vectors to a VF before SR-IOV enablement.

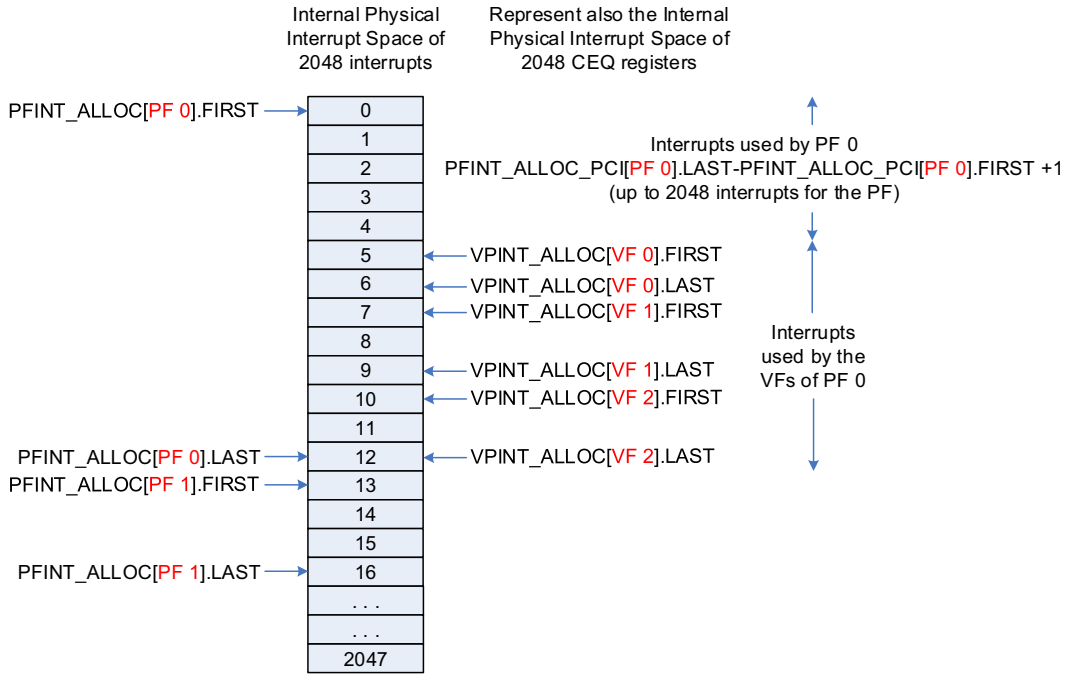


Figure 9-1. Example for Interrupt Vectors and CEQ Interrupt Control Registers Mapping to the Internals Space

9.1.1.1 Interrupt Enable Procedure

Interrupts are enabled at three levels:

- Enablement on the PCIe interface by PCI configuration registers programmed by the OS.
  - Legacy INTA/INTB/INTC/INTD interrupt message is controlled by the *Interrupt Disable* flag in the Command Register in the PCI configuration space per PF.
  - MSI is enabled by the *MSI Enable* flag in the MSI Capability structure in the PCI configuration space per PF.
  - MSI-X is enabled by the *MSI-X Enable* flag in the MSI-X Capability structure in the PCI configuration space per PF and per VF, and further enablement by the *Mask* bit per MSI-X vector in the MSI-X Table Structure.
- Enablement of the interrupts by the driver by the *INTENA* flag in the *xx*INT_DYN_CTL0 and *xx*INT_DYN_CTLN registers (where *xx*=VF), or GLINT_DYN_CTL for PF.
  - The software driver sets the *INTENA* flag and clears the *WB_ON_ITR* flag to enable the relevant interrupt signal. Upon interrupt assertion, the *INTENA* flag and, in case of legacy interrupts, the interrupt level are auto-cleared.
  - Enablement of the interrupts per cause by the *CAUSE_ENA* flag in cause control registers (see Section 9.1.2).
- Interrupt moderation:
  - Interrupt Throttling (ITR) is described in Section 9.1.4.1.
  - Interrupt rate limiting (INTRL) is described in Section 9.1.4.2.

9.1.1.2 Pending Interrupt Array - PBA

On top of the interrupt signaling on the PCIe bus, the E810 also supports the standard PBA structure in the MSI-X BAR (see BAR description in Section 14.2.6.1). The PBA is relevant only when MSI-X is enabled. It is described as part of the MSI-X Capability structure in Section 14.3.3 for the PF, and Section 14.5.3.1 for the VFs. A bit in the PBA is set to one when an interrupt is triggered internally, and cleared when the MSI-X vector is sent on the PCIe bus.

9.1.1.3 Interrupt Sequence

This section describes the interrupt sequence of events started by an internal event that triggers an interrupt until it is sent to the PCIe bus, and the expected software response. Note that the description of the interrupt sequence refers to flags that are explained in the following sections.

MSI and MSI-X interrupts while interrupts are enabled:

1. Any of the interrupt causes has an event that sets an internal *INTEVENT* flag for the matched interrupt signal.
2. If the interrupt is enabled by *INTENA* and also enabled by the interrupt moderation policy (ITR and rate limiting), the hardware executes the following steps in order:
   a. Scan all queues associated with this interrupt by a linked list explained in Section 9.1.3. Write back the status of all completed descriptors that were not reported so far, and clear the internal *EVENT* flags of these queues.
   b. Set the matched bit in the pending interrupt block array (PBA).
   c. The *INTEVENT* and *INTENA* are auto-cleared.
3. If the interrupt is enabled by the OS (by PCIe setting), the interrupt message is sent to the PCIe.
   - The interrupt indication in the PBA is auto-cleared.
4. During the interrupt handler, the software processes each individual interrupt cause.
5. At the end of the interrupt handler, the software re-enables the interrupts by setting *INTENA*.
   - On the same register, the software also sets the *CLEARPBA* flag, which clears the matched bit in the PBA.
   - On the same register, the software can update one of three ITRs of this interrupt by setting the *ITR_INDX* and *INTERVAL* fields. Setting the *ITR_INDX* to 11b does not impact the ITRs. See the ITR explanation in Section 9.1.4.1.

MSI/MSI-X interrupts while interrupts are disabled by the OS:

1. Same as above.
2. Same as above.
3. The OS polls the PBA and schedules the interrupt handler
4. Same as above.
5. Same as above.

Legacy interrupts while interrupts are enabled by the OS (mapped to interrupt zero of the PF):

1. Same as above.
2. Same as above.
3. If the interrupt is enabled by the OS (by PCIe setting), the interrupt message is sent to the PCIe.
4. At the very beginning of the interrupt service routine, the software driver clears the internal PBA by setting the *CLEARPBA* flag in the relevant GLINT_DYN_CTL register.
   - As a result, an interrupt de-assertion message is sent on the PCIe bus.

   Note: When several PFs are assigned to the same legacy interrupt, the interrupt de-assertion message must be sent only when all PFs associated with that interrupt have cleared the internal PBA as mentioned above.
5. During the interrupt handler, the software processes each individual interrupt cause.
6. At the end of the interrupt handler, the software re-enables the interrupts by setting *INTENA*.
   - On the same register, the software can update one of three ITRs as explained above.

9.1.2 Interrupt Causes

This section lists all interrupts causes (sources), while mapping these causes to the interrupt vectors is described in Section 9.1.3. Note that only the PF has access to any of the cause registers listed in this subsection. Therefore, VF is required to request its cause-mapping programming from the PF by admin command or any other sideband channel out of the scope of this document.

9.1.2.1 LAN Transmit Queues

The E810 supports 2K LAN transmit queues in legacy mode and up to 16K queues in comm mode for the whole device, where each queue is a potential interrupt cause. A status reporting of a completed descriptor with *EOP* bit set is considered as a transmit "event" that can trigger an interrupt (if the interrupt is enabled).

LAN transmit queue 'n' is enabled for interrupts by the *CAUSE_ENA* flag in its QINT_TQCTL[n] register. The queues are mapped to any interrupt vector within the function space by the *MSIX_INDX* field in the QINT_TQCTL[n] register, and mapped to any of it's ITRs (or immediate interrupt) by the *ITR_INDX* field in this register. See ITR description in Section 9.1.4.1.

9.1.2.1.1 Transmit Descriptors Write-Back and Interrupts

Completed transmit descriptors are posted to host memory or its Completion Queue (if applicable) in one of the following cases:

- Unrelated events to interrupts.
  - Transmit descriptors with active *RS* or *RE* flags are completed.
  - Completion Queue cache line contains an indication of 16 completed transmit descriptors.
- Interrupt scheduling impact on transmit descriptor write-back is a function of the *NoExpire* flag in the queue context, plus the *CAUSE_ENA* bit in the QINT_TQCTL register, plus the *INTENA* settings in the *xx*INT_DYN_CTL*x* register for the interrupt, as presented in Table 9-1.

Notes:

1. The *NoExpire* flag is expected to be set at queue initialization and remain static.
2. All transmit queues associated with the same Completion Queue must have the same *NoExpire* setting.

Table 9-1. Tx-Queue Interrupt Setting

NoExpire[1]	CAUSE_ENA	INTENA	WB_ON_ITR	Functionality[2]
N/A	0	N/A	N/A	Transmit queue is not associated with any interrupt. Descriptors write-back do not affect and are not affected by the interrupt logic. This setting cannot be used with queues associated with a Completion Queue (since nothing triggers the flush of Completion Queue).
0	1	1	0	Default LAN driver setting. Completed descriptors with EOP trigger the ITR. Completed descriptors with *RS* or *RE* flags are written back. At ITR expiration, the most recent completed descriptors with EOP is WB for all queues associated with this interrupt. Then, if Completion Queue(s) are used, its cache line is posted to host memory. Then the interrupt is initiated.
0	1	Any	1	Polling driver setting. Same as above while interrupt is NOT initiated.
1	1	1	0	This setting is used only if Completion Queues are used. Flow is the same as above, while following ITR expiration only the most recent completed descriptors with EOP are written back for ONLY one Tx-Queue per Completion Queue, and then the Completion Queue's cache line is posted to host memory. Then the interrupt is initiated.
1	1	Any	1	Same as above while interrupt is NOT initiated.

1. The "NoExpire" setting for Tx-Queues is only applicable for Completion Queues usage.
2. See Section 9.1.4.1 for details on ITR.

9.1.2.2 LAN Receive Queues

The E810 supports 2K LAN receive queues for the whole device, where each queue is a potential interrupt cause. A DMA completion of a descriptor with an *EOP* flag and its buffer is considered as a receive "event" that triggers an interrupt (if the interrupt is enabled). Furthermore, if the number of free descriptors on the receive queue drops below threshold, it is considered as "immediate event" (described in the following subsections). The low threshold is defined by the *LRXQTRESH* parameter in the receive queue context.

Similar to the LAN Transmit Queues, the LAN Receive Queues are mapped to any ITR of any interrupt vector by the matched QINT_RQCTL registers.

9.1.2.2.1 Receive Descriptors Write-Back and Interrupts

Completed receive descriptors are posted to host memory in one of the following cases:

- Unrelated events to interrupts.
  - — Whole internal cache line of descriptors are completed.
  - — All completed descriptors in the internal cache are posted to host memory when the receive queue context is evicted from the cache.

— Interrupt scheduling impact on receive descriptors write-back is a function of the *NoExpire* flag in the queue context, plus the *CAUSE_ENA* in the QINT_RQCTL register, plus the *WB_ON_ITR* and *INTENA* settings in the *xx*INT_DYN_CTL*x* register for the interrupt, as in Table 9-2.

Table 9-2. Rx-Queue Interrupt Setting

NoExpire	CAUSE_ENA	INTENA	WB_ON_ITR	Functionality[1]
0	0	N/A	N/A	Receive queue is not associated with any interrupt and descriptors write-back are made only according to the internal cache policy (in other words, when a full cache line is available for write-back).
0	1	1	0	Completed descriptors are posted to host memory according to the internal descriptor cache policy (in other words when a full cache line is available for write-back). Completed descriptors also trigger the ITR. Following ITR expiration, all leftover completed descriptors are posted to host memory, and then the interrupt is triggered. ***Note:*** This means that in case the ITR expires when the relevant cache line is not full, the same cache line is written back twice (once when the ITR expires and once when the cache line becomes full).
0	1	Any	1	Same as above but the interrupt signal is NOT triggered.
1	0	N/A	N/A	Each completed descriptor is written immediately to host memory. No interrupts are associated with the queue.
Else				Reserved. Forbidden setting options.

1. See Section 9.1.4.1 for details on ITR.

Note: Software can dynamically switch between WB_ON_ITR mode and ITR mode by setting/clearing the *WB_ON_ITR* flag.

Warning: The NoExpire mode specified in Table 9-2 can cause excessive PCIe transaction traffic that is not limited by any rate control/throttling mechanism. It is advised to enable it only for traffic types requiring low latency, and with consideration to the packet rate supported by the platform in this mode.

9.1.2.3 Protocol Engine Queues

A completion of an Asynchronous Event Queue Entry or a Completion Event Queue Entry is considered as an "event" that triggers an interrupt (if the interrupt is enabled). PE Completion Event Queues (CEQs) and Asynchronous Event Queues (AEQs) can be mapped to any interrupt, with optional interrupt moderation described in the following subsections.

The E810 supports a single Asynchronous Event Queue interrupt control register (per function). It is enabled for interrupts by the *CAUSE_ENA* flag and is mapped to any ITR of any interrupt vector by the *xx*INT_AEQCTL register (where *xx*=PF or VP).

Completion Event Queues (per function) are enabled for interrupts by the *CAUSE_ENA* flag and are mapped to any interrupt vector and its ITRs by the GLINT_CEQCTL registers. The GLINT_CEQCTL registers are allocated to the functions by the PFINT_ALLOC and VPINT_ALLOC registers (the same as the interrupts allocation). See Figure 9-1 that illustrates an example of an allocation.

Note: All PFs and up to 32 VFs can be enabled for the PE. Each enabled function has a single AEQ and a set of global CEQs. All the functions have the a full set of *xx*INT_AEQCTL and GLINT_CEQCTL registers (total of 264 AEQ interrupt control registers and 2048 CEQ interrupt control registers).

A CEQ[n] of a function is mapped to the GLINT_CEQCTL[k]. It is expected that the software uses only those registers that represent "real" AEQs and CEQs behind them.

9.1.2.3.1 CEQ Write-Back and Interrupts

Completed CEQ descriptors are posted to host memory by a similar scheme to the receive descriptors write-back policy. It depends on the settings of the *NoExpire* flag in the TBD register, plus the *CAUSE_ENA* in the GLINT_CEQCTL, plus the *WB_ON_ITR* and *INTENA* settings in the *xx*INT_DYN_CTL*x* register for the interrupt.

9.1.2.4 Admin Queues

The E810 supports three types of Control Queues: Firmware Admin Queues (supported also in predecessor devices), PF/VF Mailbox queues, and Sideband queues.

A completion of an admin command on the transmit queue or posting a structure on the receive queue is considered an event for the queue pair that can trigger an interrupt (depending on if it was configured to do so in the relevant descriptor). The registers that map these queue pairs to interrupt vectors are listed in the following table:

Control Queue Pairs	Mapping to Interrupt Registers
PF Mailbox queues	• PFINT_MBX_CTL maps the PF/VF mailbox of PF0...PF7 to its interrupts. • PF0INT_MBX_CPM_CTL maps the CPM PFVF MBX to its interrupt. • PF0INT_MBX_HLP_CTL maps the HLP PFVF MBX to its interrupt. • PF0INT_MBX_PSM_CTL maps the PSM PFVF MBX to its interrupt.
VF Mailbox queues	• VPINT_MBX_CTL maps the PF/VF mailbox of VF0...VF767 to its interrupts. • VPINT_MBX_CPM_CTL maps the CPM PFVF MBX of VF0...VF127 to its interrupts. • VPINT_MBX_HLP_CTL maps the HLP PFVF MBX of VF0...VF15 to its interrupts. • VPINT_MBX_PSM_CTL maps the PSM PFVF MBX of VF0...VF15 to its interrupts.
PF Sideband queues	• PFINT_SB_CTL maps the SB-IOSF mailbox of PF0...PF7 to its interrupts. • PF0INT_SB_CPM_CTL maps the CPM SB-IOSF MBX to its interrupt. • PF0INT_SB_HLP_CTL maps the HLP SB-IOSF MBX to its interrupt.
Firmware Admin Queues: PF, HLP, and PSM AQs	• PFINT_FW_CTL map the firmware AQ of PF0...PF7 to its interrupts. • PF0INT_FW_HLP_CTL maps the HLP FW AQ to its interrupts. • PF0INT_FW_PSM_CTL maps the PSM FW AQ to its interrupts.

9.1.2.5 Other Interrupt Causes

The E810 supports asynchronous events that can generate interrupts for the PFs. The "other" causes interrupt events supported by the PFs are indicated by the PFINT_OICR registers and enabled by the PFINT_OICR_ENA registers per PF. The register's fields are shown in the Table 9-3.

The "other" causes are mapped to an interrupt vector per PF by the PFINT_OICR_CTL registers.

Note: The "other" causes must be mapped to an interrupt vector index that is within the boundaries of the allocating PF and cannot be mapped to an interrupt vector allocated to any of the VFs. Reusing the example given in Figure 9-1, for PF #0, the OICR can only be mapped to interrupt vector index in the range [0..4], which are the only vectors in PF #0 range that are not mapped to any VF.

Any "other" cause event sets its matched bit in the PFINT_OICR register. If enabled by the PFINT_OICR_ENA register, it triggers an interrupt defined by the PFINT_OCTL register.

During nominal operation it is expected that the software reads the PFINT_OICR register that indicates the "other" cause events (this register is read/clear). Setting the flags in the PFINT_OICR register (other than the queue flags and the *SWINT* flag), emulates an interrupt event of the specific cause (if enabled by the PFINT_OICR_ENA register).

Note: Other causes are latched by the hardware even when the interrupt is not initialized or enabled by the software. Therefore, when software first enables the interrupt, the hardware can trigger an interrupt of events that occurred while the interrupt was not enabled. If software wishes to avoid such behavior, it can read and clear the other causes CSR PFINT_OICR before enabling the interrupt. This behavior is changed from previous generations.

Table 9-3. "Other" Interrupt Causes of the PFs

"Other" Cause	Description	Affected Functions
Queue	Any LAN or PE queues linked to the other cause interrupt (with the exception of an AEQ).	
TSYN_TX	Tx packet time is sampled in the PHY(s).	Indicated PFs according to port to PF mapping (configured by PFGEN_PORTNUM) for ports 0…7 and to PF0 for ports 8 and above.
TSYN_EVNT	Event is sampled by the 1588 timer due to a transition in one of the 1588 input GPIOs.	Indicated PFs according to the *PF_MASTER* field in the GLINT_TSYN_PFMSTR registers.
TSYN_TGT	One of the target time of the 1588 timer is expired.	Same as TSYN_EVNT.
ECC_ERR	Unrecoverable ECC Error. This bit is set when an unrecoverable error is detected in one of the device memories.	Reported to all PFs.
MAL_DETECT	Malicious programming detected.	Reflected to the parent PF of the malicious VF.
GRST	Global Resets Requested (CORER, GLOBR or EMPR).	Reported to all PFs.
GPIO	GPIO Event indicates an event on any of the GPIO pins enabled for interrupt by the PFINT_GPIO_ENA register. The GPIO state can be fetched on the GLGEN_GPIO_STAT register. The level transition that generates an interrupt is set for GPIO 'n' by the *INT_MODE* field in the matched GLGEN_GPIO_CTL[n] register.	Reported to the relevant PF
STORM_DETECT	Indicates a change entering the storm control state of the LAN port that is connected to this PF. The storm control state is reflected in the PRT_SWT_SCSTS register.	Reflected to the PF associated with the port.
HMC_ERR	HMC errors: PEPMAT or FPMAT. Specific PEMAT errors are reported in the PFHMC_ERRORINFO and PFHMC_ERRORDATA registers. Specific FPMAT errors are reported in the PFHMC_ERRORINFO_FPMAT and PFCHMC_ERRORDATA_FPMAT registers.	Reported to the relevant PF.
PE_CRITERR	Protocol Engine Critical Error. Indicates that the Protocol Engine has encountered a critical error.	Reported to all PFs.
VFLR	VFLR was initiated by one of the VFs of the PF. The PF software should read the GLGEN_VFLRSTAT getting an indication for the VF that generated the VFLR.	Reported to the parent PF.
VFR_DONE	VFR sequence by the hardware is completed. The PF software should read the GLGEN_VFRDONE indicating it to the VF driver.	Reported to the parent PF.
SWINT	Software interrupt (detailed in Section 9.1.2.6.	Reported to the PF that generated the interrupt.
PE_PUSH_OVERFLOW	Indication that the PE push buffer is overflowing.	Reported to all PFs.

9.1.2.6 Software Initiated Interrupt

In some cases, the software might not be able to process all events in a single interrupt handler. In such cases, the software can schedule another interrupt to complete processing all interrupt events. The software can trigger an interrupt on any vector and any of its ITRs or NoITR. The software interrupt is mapped to one of three ITRs or immediate interrupt by the *SW_ITR_INDX* field in the *xx*INT_DYN_CTL*x* registers (where *xx*=PF or VF and x=0 or N). When programming the *SW_ITR_INDX* parameter, the *SW_ITR_INDX_ENA* flag in this register should be set as well.

The software initiates the interrupt by setting the *SWINT_TRIG* flag in the *xx*INT_DYN_CTL*x* registers.

Software has the option of setting the *SWINT_TRIG* flag in the *xx*INT_DYN_CTL*x* registers with or without setting the *INTENA* flag by using the *INTENA_MSK* flag in the same register. Setting or clearing the *INTENA* flag, not as part of an interrupt handling routine, can lead to race conditions. Therefore, it is expected that software never clears the *INTENA* flag, and clearing of the *INTENA* flag is always done by hardware for non-debug conditions. Also, it is expected that software sets *SWINT_TRIG* together with *INTENA* only as part of the interrupt handling routine (in other words, at the end of handling an interrupt).

Setting the software interrupt in the vector of the PF that is assigned for the other causes, the software *INT* flag is set in the OICR register of that PF (on top of the interrupt triggering).

9.1.2.7 Interrupt Status Registers

During nominal operation, the software avoids any possible read accesses. The other causes interrupts of the PFs is an exception. Following an other causes interrupt, the software is expected to read the OICR register identifying the source of the interrupt. The OICR register provides indication for the following events:

- Any of the "other" interrupt causes.
- An indication for LAN receive or transmit queues CEQs that might be associated with the OICR interrupt.
- SWINT indication, which is a result of software initiated interrupt.

9.1.3 Interrupt Linked List

The queues are linked to a specific interrupt vector by the MSIX_INDX field in the *xxx*QCTL registers. Specifically, the LAN queues and the PE completion event queues are associated with an interrupt vector by the following registers: QINT_RQCTL, QINT_TQCTL, and GLINT_CEQCTL. Once an event is initiated on any of these queues, it is added to the linked list of its associated interrupt (unless the *NoExpire* flag is set for the event). Interrupt causes mapping to the interrupt vectors are illustrated in Figure 9-2.

When an interrupt sequence is triggered, the E810 process all active queues in the linked list of this interrupt. While processing the queues in the linked list, the hardware writes back the status for those completed descriptors that were not reported already, as listed below:

- **LAN Receive Queues** — The hardware triggers a status write-back of all completed descriptors.
- **LAN Transmit Queues (in legacy mode)** — The hardware writes back the completion indication of the last completed transmit descriptor that has an *EOP* flag (regardless of the *RS* bit) and was not already reported to host memory.

- **LAN Transmit Queues (in comm mode)** — The hardware writes back the completion indication of the last completed transmit descriptor that has an *EOP* flag (regardless of the *RS* bit) and was not already reported to its Completion Queue. Once a whole cache line of 64 bytes of the Completion Queue is written internally, it is posted to host memory. When all Tx-Queues in the interrupt linked list that are associated with a specific Completion Queue are processed, any remaining cache line residual is posted to host memory as well.

  Note: Multiple Completion Queues can be assigned to the same interrupt. Yet, all transmit queues that are associated with a specific Completion Queue must be linked by the software to the same interrupt.

- **PE CEQ entries** — The hardware writes back the completion indication of all completed CEQ entries.

Note: The *xxx*QCTL registers are accessible only to the PF. A VF can assign causes to interrupts with the help of the PF using an admin command or any other sideband message (out of the scope of this document).

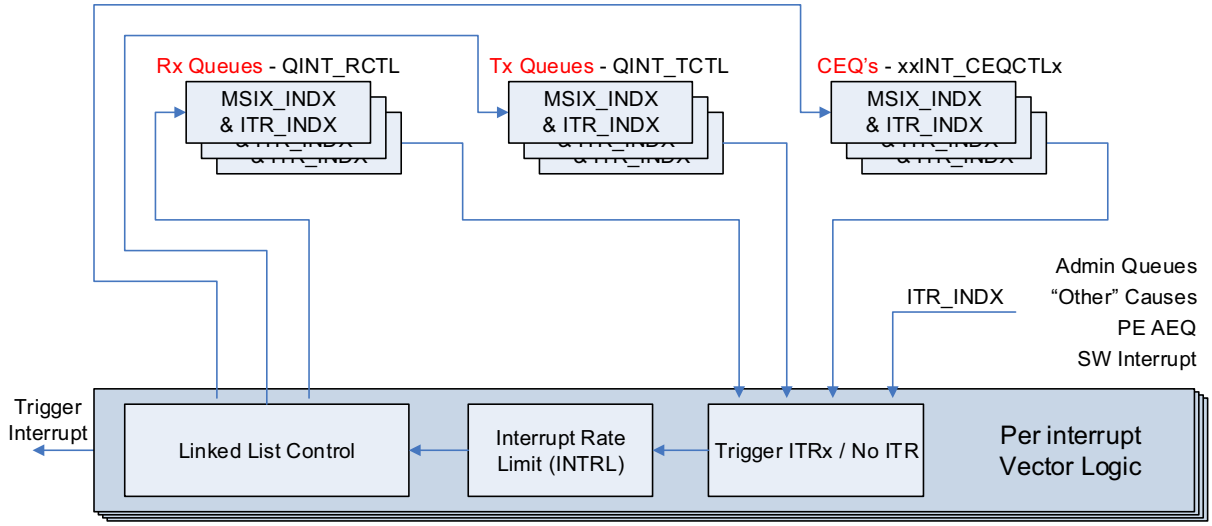


Figure 9-2. Mapping Interrupt Causes to Interrupt Signaling

9.1.3.1 Interrupt Linked List Management

This section describes the required software flow for adding and removing an interrupt cause from an interrupt linked list.

9.1.3.1.1 Initial Linked List Setting and Adding Interrupt Causes

Any interrupt cause can be added to an interrupt vector at any time (either before or after the interrupt is already active). The interrupt causes are linked to an interrupt vector by setting the *MSIX_INDX* field in the *xxx*QCTL register of the cause. These registers also map the cause to a specific *ITR_INDX* of the interrupt and also contain the interrupt *CAUSE_ENA* flag.

9.1.3.1.2 Removing an Interrupt Cause from an Active Interrupt

An interrupt cause (queue) can be removed from an active interrupt during run time according to the following steps. This flow is needed as part of a queue disable flow or a VM reset.

1. Disable the interrupt for the cause by clearing the *CAUSE_ENA* flag in the *xxx*QCTL register of the cause.
2. Generate a software interrupt on the same vector.
3. After the interrupt is generated, it is safe to disable the queue or initiate the VM reset.

Note: This flow is not required for VF or PR resets or any higher reset source.

9.1.3.1.3 Migrating a Queue Between Interrupt Vectors

Migrating a queue (or other cause) from one interrupt to another one is done as follows:

1. Software remaps the queue by simply setting of the *MSIX_INDX* field to the new interrupt, together with the *ITR_INDX* and *CAUSE_ENA* flag in the *xxx*QCTL register of the cause.

   Note: Depending on timing, the previous interrupt could still be initiated for the queue even after this action.

2. To ensure that the original interrupt vector is no longer triggered by events associated with the migrated queue, software should trigger a software interrupt to force the processing of all causes associated with the original interrupt.

Note: Transmit queues associated with a Completion Queue cannot be migrated dynamically. Migrating a queue in this case violates the rule by which all transmit queues of a specific Completion Queue must belong to the same interrupt.

9.1.4 Interrupt Moderation

The E810 is able to throttle interrupts in two layered methods:

- Interrupt Throttling (ITR)
- Interrupt Rate limiting (INTRL).

These methods are detailed in the following subsections.

9.1.4.1 Interrupt Throttling (ITR)

Interrupt throttling is a mechanism that guarantees a minimum gap between two consecutive interrupts (other than possible jitter caused by handling the interrupts). The E810 counts the time since the last interrupt is scheduled and compares it against the ITR setting. If an event associated with this ITR happens before the ITR expires, the interrupt assertion is delayed until the ITR expires. If the ITR expires before any event associated with this interrupt, the interrupt logic is "armed" and the interrupt can be asserted the moment the event happens. The ITR intervals per vector are programmed by the *xx*INT_ITR*x* registers. The ITR is measured in units corresponding to the maximal aggregated port speeds allowed by the device (see Section 9.1.4.3). Note that ITR expiration sequence is triggered only when all the following conditions are met:

- The ITR timer is expired.
- The *INTENA* or the *WB_ON_ITR* flags in the matched GLINT_DYN_CTL register is set.

- The interrupt has credit(s) by the "Interrupt rate limiting" logic explained in Section 9.1.4.2.

Additionally, to the terms mentioned above, the ITR expires when the interval setting for that ITR is changed.

The E810 supports three ITRs per MSI-X vector, as well as a NoITR option. The interrupt causes are mapped to one of the ITRs by the *ITR_INDX* field (per cause). The ITR intervals can be programmed directly to the *xx*INT_ITR*x* registers or via the *xx*INT_DYN_CTL*x* registers. It might be useful to set the initial values using the *xx*INT_ITR*x* registers and dynamic update by the *xx*INT_DYN_CTL*x* registers, as explained in Step 4 of the interrupt sequence explained in Section 9.1.1.3.

When any ITR interval of an interrupt with a pending event is expired and the INTRL[1] credit is positive, the hardware follows these steps:

1. Clear the other ITRs of the same interrupt.
2. Process all causes of the same interrupt (associated with all ITRs).

9.1.4.2 Interrupt Rate Limiting (INTRL)

Interrupt rate limiting is a credit based mechanism that limits the maximum average number of interrupts per second. The PF controls the rate limit for the vectors in its space (including his VFs) using the GLINT_RATE registers. The control parameters of these registers are detailed below.

- ***INTRL_ENA*** — Enable/Disable option for the INTRL scheme. When disabled, interrupts can be generated without rate limiting control.
- ***INTERVAL*** — The INTRL is a 6-bit interval defined in units determined by the device configuration (see Section 9.1.4.3). The value determines the time gap on which new interrupt credit is gained.

9.1.4.3 INTRL/ITR Timing Granularity

Due to the optimization of the internal clock speed in the E810 according to the total aggregated bandwidth allowed by the device (in other words, the sum of all allowed ports linked at the highest allowable speed), the timing granularity (time units) used for configuring the INTRL and ITR intervals changes in accordance with the prior. The granularity is determined according to the device's configuration during the power-on phase, and is static throughout its operation.

Table 9-4 specifies the INT and INTRL timing granularity values as a function of the relevant device configuration.

Table 9-4. ITR/INTRL Granularity Values

Aggregated Bandwidth	ITR Granularity	INTRL Granularity
Less than or equal to 25 Gb/s	4 µs	8 µs
Less than or equal to 50 Gb/s	2 µs	4 µs
Above 50 Gb/s	2 µs	4 µs

1. See Section 9.1.4.2 description of the Interrupt Rate Limiting (INTRL).

9.2 Virtualization

9.2.1 Overview

I/O virtualization is a mechanism to share I/O resources among several consumers. For example, in a virtual system, multiple operating systems are loaded and each executes as though the whole system's resources are 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 must 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 are shared. Thus, they can be accessed directly from each guest operating system or Virtual Machine (VM).

Two modes to support operation in a virtualized environment were implemented in previous products:

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 called IOV mode in this section.
2. Central management of the networking resources by an IOVM or by the VMM (also known as software switch acceleration mode). This mode is called Next Generation VMDq mode.

The E810 fully supports Next Generation VMDq mode and SR-IOV. The E810 supports two modes of offloads as part of Next Generation VMDq: VMDq1 and VMDq2.

In VMDq1, all the VMs are part of the same VSI, and each VM is allocated a single queue. There is no replication of packets to different VMs, and there is no forwarding of traffic from one VMDq1 VM to another.

In VMDq2, each VM is assigned a switch port (VSI). Thus, there can be full switching between ports, including VM-to-VM switching and replication of multicast packets.

On top of these modes, The E810 supports Scalable I/O (which is a lightweight version of SR-IOV), in which traffic and resources are assigned to VMs through an Assignable Interface (AI) that is a CSR space within the PF space exposed to the VM. Traffic belonging to an AI is identified through a PASID prefix. See Section 9.2.3 for details on Scalable I/O and PASID.

In a virtualized environment, the E810 serves up to 768 Virtual Machines (VMs) per device; 256 of these can be directly assigned and any of them can be accessed in Next Generation VMDq mode. See Section 9.2.2.4 for details of the VFs and VMs resource allocation.

Most configurations and resources of the device are shared across VMs. The PF driver must resolve any conflicts in configuration between the VMs. For example, the PF driver should manage all the link configuration requests of the VFs.

Most of the virtualization offload capabilities provided by the E810, apart from the replication of functions defined in the PCI-SIG IOV specification, are also part of Next Generation VMDq.

A hybrid model, where some of the virtual machines are assigned a dedicated share of the port and the others are serviced by an IOVM, is also supported. This model can be used when some of the VMs run operating systems for which VF drivers are available. Such configurations can benefit from IOV. Others can run older operating Systems for which VF drivers are not available, and are serviced by an intermediary or models where VFs are assigned to VMs requiring a higher networking bandwidth. In this last case, the IOVM or VMM is assigned some VSIs and receives all the packets with MAC Addresses of the VMs behind it. VSIs are described in Section 7.8.8.

The following section describes the support that the E810 provides for virtualization. This section assumes a single-root implementation of IOV and no support for multi-root.

9.2.1.1 Direct Assignment Model

The direct assignment support in the E810 is built according to the software model defined by the SR-IOV specification.

The Physical function (PF) driver is responsible for the initialization and the handling of the common resources of the port. Other drivers (VF drivers) might read part of the status of the common parts, but cannot change it. 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-Queue and Rx-Queue or interrupt handling, is replicated per VF. It is 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 this section refers to the VMM.

A channel is provided between the VF driver and the PF driver through the use of admin commands. See Section 9.5.14.

9.2.1.1.1 Rationale

The purpose of direct assignment is to enable each of the virtual machines to receive and transmit packets with minimum overhead. The non-time-critical operations (such as init 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 the 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).
- Interrupt handling.

The capabilities needed to provide independence between VMs are:

- Per VM reset and enable capabilities.
- Tx QoS control.
- Allocation of separate CSR space per VM.

The queue context creation, rate control, and VF enable capabilities are controlled by the PF.

9.2.1.2 Virtualized System Overview

The following drawings describe the various elements involved in the I/O process in a virtualized system. Figure 9-3 describes the flow in software Next Generation VMDq operation mode, while Figure 9-4 describes the flow in IOV mode.

This document assumes that in IOV mode, the driver on the guest operating system is aware that it works in a virtual system (para-virtualized) and there is a channel between each of the virtual machine drivers and the PF driver, allowing message passing (such as configuration request or interrupt messages). This channel might use the mailbox implemented in the E810 or any other means provided by the VMM vendor.

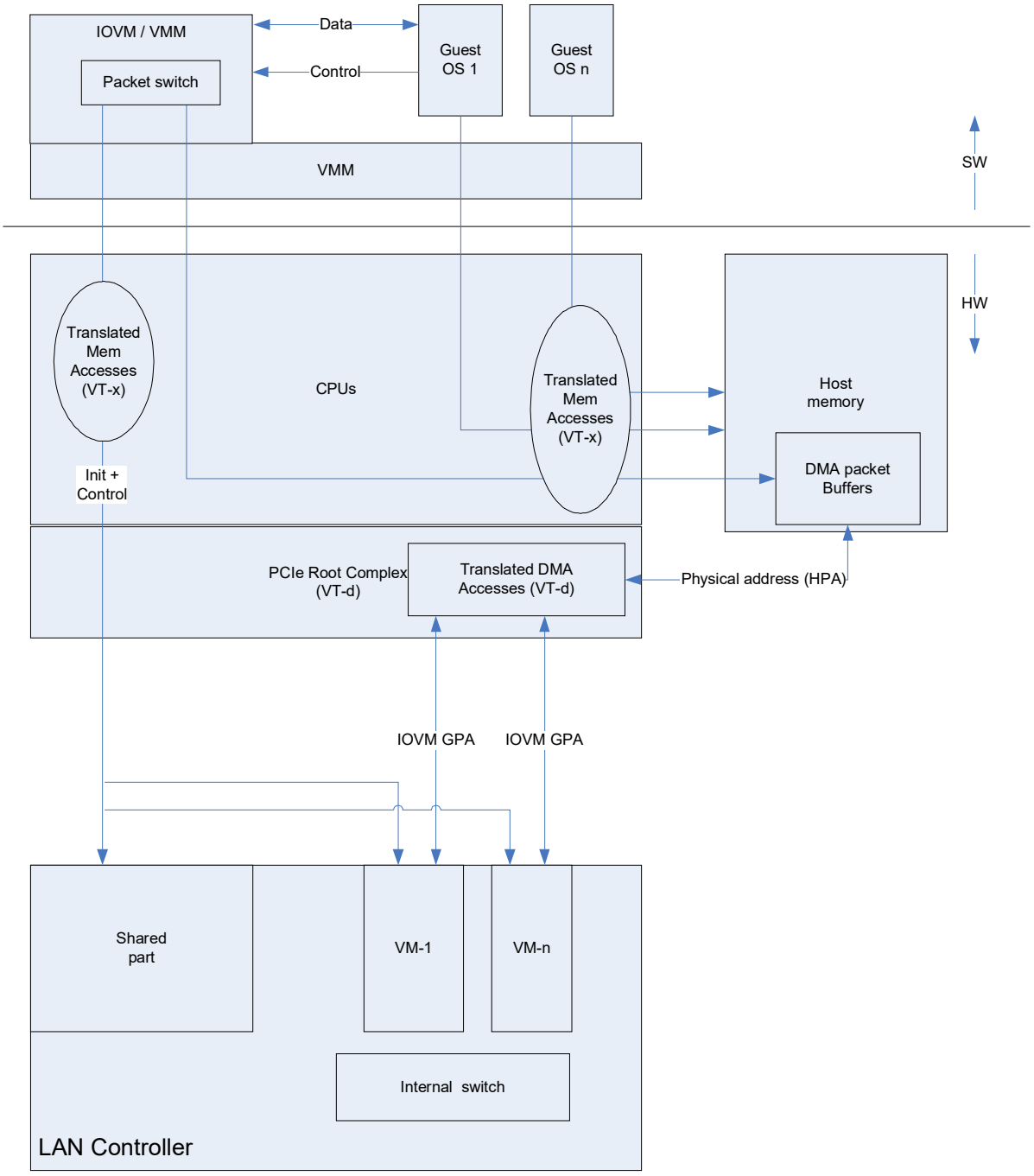


Figure 9-3. VMDq System

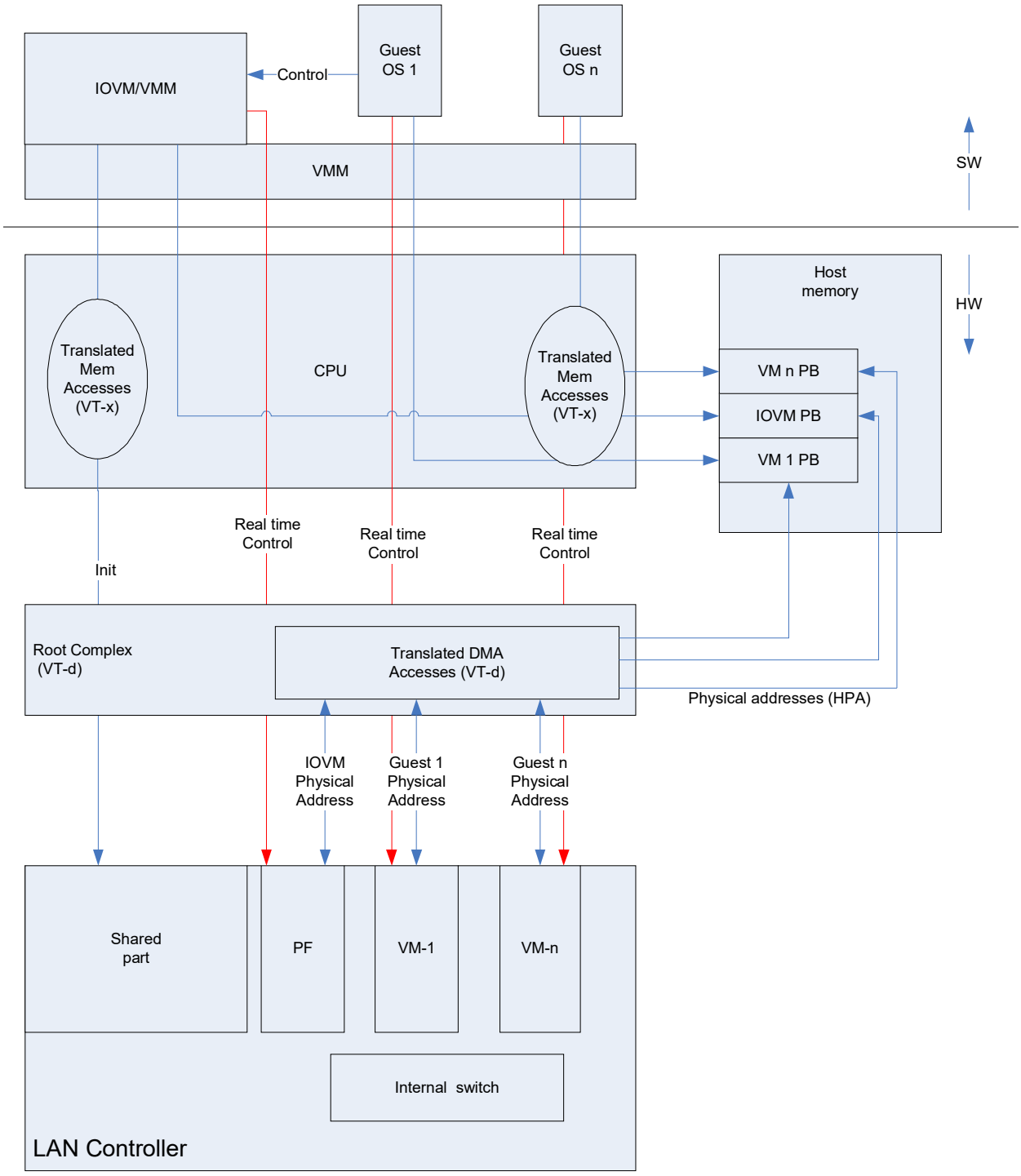


Figure 9-4. SR-IOV Based System

9.2.1.3 Virtualization Supported Features

The E810 supports a superset of the virtualization features supported in previous products. The following table compares the virtualization features of the E810 with these of the X710/XXV710/XL710.

Table 9-5. E810 Versus X710/XXV710/XL710 Virtualization Support

Feature	E810 Support	X710/XXV710/XL710 Support
Virtualization-Specific Capabilities		
SR-IOV support	Yes	Yes
VMDq1 support	Yes	Yes
VMDq2 support	Yes	Yes
ATS support	No	No
VF-to-PF mailbox	Yes (in hardware).	Yes (via firmware).
Base mode VF support	Yes. Compatible with the X710/XXV710/ XL710.	Yes.
Support for Generic Capabilities		
RSS	Yes (per VSI). Small or medium size tables.	Yes (per VSI). Small or medium size tables.
DCB	Yes	Yes
Queue selection criteria	Programmable	SA, VLAN pairs or SA or VLAN
Statistics	Per VSI	Per VSI
Stateless Offloads	Per queue	Per queue
RDMA	Supported for 32 VFs	Supported for 32 VFs
Quantities		
Max number of Virtual Machines (VMs)	768	256
Max number of Virtual functions (VFs)	256 per device (globally)	128 per device (globally)
Max number of queues per VF	16 arbitrary or 256 contiguous	16 arbitrary or 256 contiguous
Max number of queues per VMDq2 VSI	16K (Tx) + 2K (Rx)	1536
Max number of queues per VMDq1 VM	According to Hash Filter configuration	1
Max number of VMDq2 VSIs	768	256
Interrupt vectors	Up to 9 or 65 per VF	Up to 9 per VF
Switching Capabilities		
MAC Addresses	Up to switch capacity	1024 per device (globally)
VLAN tags	Up to switch capacity	512 per device (globally)
Switching modes	VEB, VEPA	VEB, VEPA
VM-to-VM switching	Yes	Yes
Broadcast and multicast replication	Yes	Yes
MAC and VLAN anti-spoof protection	Yes	Yes
VLAN filtering	Global and per pool	Global and per pool
Drop if no VIS selected	Yes	Yes
Mirroring	Yes	Yes
Promiscuous modes per VM	VLAN, multicast, unicast	VLAN, multicast, unicast

For more details about the switching support, see Section 7.8.

9.2.1.3.1 Enablement of Virtualization Features

Table 9-6 describes the way to enable each virtualization feature.

Table 9-6. Virtualization Features Enablement

Technology	Included Features	Enablement
SR-IOV		Set the GLPCI_CAPSUP.*IOV_EN* bit.
VEB	VEB, VEPA, VMDq2, mirroring	Always enabled.

9.2.2 SR-IOV Implementation

9.2.2.1 IOV Concepts

The SR-IOV specification defines the following entities in relation to I/O virtualization:

- **Virtual Machine (VM)** — A virtual machine to which I/O resources are assigned.
- **A PCIe device** — The physical device that might contain a few physical functions. In this case, the E810.
- **Physical function (PF)** — A function representing a physical instance. In this case, a PCIe function that represents a physical port or a logical port. 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 VM.

9.2.2.2 IOV Control

To control the IOV operation, the physical driver is provided with a set of registers and capabilities. These include:

- The PF_VIRT_STATUS register, which indicates whether SR-IOV is enabled and the number of VFs enabled.
- Driver-to-driver communication provided by the virtualization admin commands (see Section 9.5.14).
- Switch and filtering control admin commands (described in Section 7.8.12).
- Reset indications and traffic enables registers per VF using the GLGEN_VFLRSTAT.*VFLRS* bit, indicating that a VFLR reset occurred in one of the VFs. When the GLGEN_VFLRSTAT.*VFLRS* bit is set for a given VF, this VF cannot send or receive packets. The PF should clear this bit to enable a VF.
- Malicious driver detection (described in Section 9.2.2.2.1).

The flow used to configure a function when SR-IOV is enabled is described in Section 4.4.1.2.

9.2.2.2.1 Interrupt on Misbehavior of VM (Malicious Driver Detection)

The E810 can protect itself from faulty or malicious behavior on the part of a VM driver. This is done by checking for specific illegal events.

The bits that enable these checks are grouped into four categories: Rx checks, Tx-Descriptor checks, Tx tail bump and quanta queue checks, and Tx data checks. Individual checks are controlled by global registers GL_MDCK_RX, GL_MDCK_TCMD_TCLAN, GL_MDCK_EN_TX_PQM, and GL_TDPU_DROP_DIS, respectively. If a check is not enabled (the bit is cleared) and such an event happens, the monitoring hardware does not react to the condition, and the device might malfunction.

Table 9-7, Table 9-8, and Table 9-9 list the checks in each group, the register bits used to enable them, and the value read from the MDET registers when an event is detected.

The default values for these registers are loaded from NVM.

The checks apply both to VF and PF activity. If such behavior is detected, the queue is stopped and an interrupt is sent to the PF that owns the function and should be re-initialized. For Rx-Queues, the *Queue_Block Indication* bit (bit 21) is set in the context descriptor. This bit must be cleared as part of the queue init before it can be reused.

Note: This means that a PF will be interrupted both on events from its queues and on events from its VFs queues.

Tracking which functions have caused an event is done by reading each function's VP_MDET_RX, VP_MDET_TX_TCLAN, VP_MDET_TX_PQM, and VP_MDET_TX_TDPU registers for VFs or PF_MDET_RX PF_MDET_TX_TCLAN, PF_MDET_TX_PQM, and PF_MDET_TX_TDPU for PFs. The registers are cleared by writing 1's to them. After such an event the function needs to reset the queue. Tx Data protection is different in that it does not stop the queue; it drops the offending packet.

Four global debug registers (GL_MDET_TX_PQM, GL_MDET_TX_TDPU, GL_MDET_TX_TCLAN, and GL_MDET_RX) record the function number event ID and queue number for the first event observed on Tx and Rx, respectively. These registers are cleared by writing 1's.

Note: For Tx events (GL_MDET_TX_PQM, GL_MDET_TX_TDPU, GL_MDET_TX_TCLAN), the *VF_NUM* field in the global registers can indicate the VF, the VM or if 0, an error in a PF owned VSI. The software device driver should deduct the actual offending agent from the reported queue number and the reporting in the per agent registers. For Rx events (GL_MDET_RX), only the queue number is valid. The offending function should be derived from the owner of the offending queue.

Since the malicious driver event indication in the various Tx registers is per function, and the details of the event in the GL_MDET_TX_PQM, GL_MDET_TX_TDPU, and GL_MDET_TX_TLAN registers are common to all functions, it might not be simple to determine which queue caused the event. Therefore, the driver might elect to reset the whole function. The PRTDCB_TDPUC contains information per port for the first malicious event that was detected.

9.2.2.2.1.1 Tx Data Checks (GL_MDCK_TX _TDPU Register)

The list below describes the checks that are done by the E810 on data fetches for Tx data:

- TTL error
- Anti-spoof fail[1]
- PCIe unsupported request (wrong buffer address)
- Malicious offset detected

1. During a function level reset (PFR, FLR, VFR, VMR), spurious malicious events due to anti-spoof event might occur.

- Malicious command detected
- Packet size bigger than allowed
- L2 acceptance failed
- DSI packet requested
- Malicious IPsec offload detected
- DSCP enforcement - packet is blocked

9.2.2.2.1.2 Tx Descriptor Validity Checks (GL_MDCK_TCMD Register)

Table 9-7 describes the checks that are done by the E810 on Tx-Descriptor.

Table 9-7. Malicious Driver - Tx-Descriptor Checks (GL_MDCK_TDAT_TCLAN)

Index	Event	Mode[1]	Comments
0	Wrong order/Format of descriptors	BC+COMS	• Wrong Descriptor type. • Wrong order/format of descriptors. • DSI traffic with TSO request. • Non data buffer in the middle of a packet (before EOP arrived). • Wrong descriptor type. • Non NOP descriptor arrives after TLEN reached and before total_descs_in_tso reached. ***Note:*** A null buffer triggers an event even if this bit is cleared.
1	Unsupported Requests	BC+COMS	Descriptor fetch failed.
2	Tail descriptor is not DDESC with EOP/NOP	BC+COMS	
3	False Scheduling	BC+COMS	tail == head (No new descriptors)
4	Tail value is bigger than ring length	BC+COMS	Non endless transmit mode.
5	More than 8 data commands in packet	BC+COMS	
6	Zero packets sent in quanta and no head update in this quanta	BC+COMS	
7	Packet too small or packet too big	BC+COMS	LSO mode: Check that headers + MSS/Last Segment size <= GLCOMM_MIN_MAX_PKT.*MAHDL*. Check that headers + MSS/Last Segment size >= GLCOMM_MIN_MAX_PKT.*LSO_COMS_MIHDL* (COMS mode). Check that headers + MSS/Last Segment size >= GLCOMM_MIN_MAX_PKT.*MIHDL* (BC mode). SSO Mode: Check that packet <=GLCOMM_MIN_MAX_PKT.*MAHDL*. Check that packet >= GLCOMM_MIN_MAX_PKT.*MIHDL*.
8	TSO: TLEN is not coherent with sum {LSO buffers}	BC+COMS	DataDesc.EOP arrives before TSO data ended. TLEN ended and DataDesc.EOP==0. TLEN ended, data descriptor buffer is not fully used.
9	TSO: Tail reached before TLEN ended	BC+COMS	
10	TSO: Headers are spread on more than 3 descriptors	BC+COMS	
11	TSO: Sum of TSO buffers < sum of headers	BC+COMS	EOP on header descriptor.

Table 9-7. Malicious Driver - Tx-Descriptor Checks (GL_MDCK_TDAT_TCLAN) [continued]

Index	Event	Mode[1]	Comments
12	TSO: Sum of headers is 0/MSS is 0/TLEN is 0	BC+COMS	Iplen==l4len==maclen == 0 Mss == 0 Tlen == 0 COMS:total_descs_in_lso == 0
13	TSO: CTX descriptor wrong IPSEC fields	BC+COMS	CDESC.cipherblk == 1 or > 3 | CDESC.ICVLEN >1000
14	SSO: Quanta does not include a whole number of SSO packets	COMS	Quanta must finish with NOP/Ddesc.EOP. CDESC with TSO bit set is not allowed.
15	SSO+TSO: Quanta bytes before additions exceed pkt_len*64	COMS	SSO LSO last quanta
16	SSO+TSO: Quanta commands exceed max_cmds_in_sq	COMS	Can be disabled by CSR (sometimes it can happen).
17	TSO: total_descs_in_lso is not coherent with last_lso_quanta	COMS	total_descs_in_tso reached and last_lso_quanta not reached (is 0). last_lso_quanta reached (== 1) and total_descs_in_tso not reached.
18	TSO: total_descs_in_lso is not coherent with TLEN	COMS	total_descs_in_lso descriptors processed but TLEN was not reached.
19	TSO: Quanta bytes is spread on more than max descriptors in quanta	COMS	last_quanta_in_lso = 0, quanta max descriptors finished, and (pktlen-1)* 64 bytes were not reached.
20	Number of packets in quanta mismatch	COMS	Number of sent packets in quanta does not equal number of packets in scheduling command.

1. BC - Backward Compatible. COMS - uses quanta queue.

9.2.2.2.1.3 Tx Checks (GL_MCK_EN_TX_PQM)

The following checks are done on the tail bump events and quanta queues:

Table 9-8. Malicious Driver - Tx-Descriptor Checks (GL_MDCK_EN_TX_PQM)

Index	Event	Comments
0	PCI_DUMMY_COMP	Enable detection of PCI Dummy Completion (for a QD Fetch Request).
1	PCI_UR_COMP	Enable detection of PCI "Unsupported Request" Completion (for a QD Fetch Request).
3	RCV_SH_BE_LSO	Empty Q fetch (initiated by PQMMNG after 1st Quanta was delivered to PQMMNG by DBL) and LSO QD is expected in completion. Completion is received however received QD is NOT LSO.
4	Q_FL_MNG_EPY_CH	In fetch command (initiated by PQMMNG) force_fetch is set (Q is full PQMMNG wise) but PQMQDC concluded that the Q is empty.
5	Q_EPY_MNG_FL_CH	In fetch command (initiated by PQMMNG) force_fetch is clear (Q is empty PQMMNG wise) but PQMQDC concludes that the Q is not empty.
6	LSO_NUMDESCS_ZERO	Enable detection of LSO QD whose Number of Descriptor is zero.
7	LSO_LENGTH_ZERO	Enable detection of LSO QD whose Length is zero.
8	LSO_MSS_BELOW_MIN	Enable detection of LSO QD whose MSS is below GL_MDCK_CFG2_TX_PQM.*LSO_MIN_MSS*.
9	LSO_MSS_ABOVE_MAX	Enable detection of LSO QD whose MSS is above GL_MDCK_CFG2_TX_PQM.*LSO_MAX_MSS*.
10	LSO_HDR_SIZE_ZERO	Enable detection of LSO QD whose Header Size is zero.
11	RCV_CNT_BE_LSO	Enable detection of LSO QD for a Q which is LSO disabled.

Table 9-8. Malicious Driver - Tx-Descriptor Checks (GL_MDCK_EN_TX_PQM) [continued]

Index	Event	Comments
12	SKIP_ONE_QT_ONLY	The 1st Quanta of a LSO QD which is composed of multiple Quantas is delivered to hardware. The 1st Quanta is scheduled and hardware fetches LSO QD so it can schedule the other Quantas of the same LSO QD. During processing of the fetched LSO QD, hardware concludes it is a single Quanta (which already been scheduled) although doorbell suggested it is a multiple Quantas.
13	LSO_PKTCNT_ZERO	Enable detection of LSO DBL whose first Quanta Number of Segments is zero
14	SSO_LENGTH_ZERO	Enable detection of SSO QD whose Length is Zero.
15	SSO_LENGTH_EXCEED	Enable detection of SSO QD whose Length exceeds GL_MDCK_CFG1_TX_PQM.*SSO_MAX_DATA_LEN*.
16	SSO_PKTCNT_ZERO	Enable detection of SSO QD whose Packet Count is Zero.
17	SSO_PKTCNT_EXCEED	Enable detection of SSO QD whose Packet Count exceeds GL_MDCK_CFG1_TX_PQM.*SSO_MAX_PKT_CNT*.
18	SSO_NUMDESCS_ZERO	Enable detection of SSO QD whose Number of Descriptor is zero.
19	SSO_NUMDESCS_EXCEED	Enable detection of SSO QD whose Number of Descriptor exceed Minimum of {Remain-Ring-Length and GL_MDCK_CFG1_TX_PQM.SSO_MAX_DESC_CNT}.
20	TAIL_GT_RING_LENGTH	Enable detection of DBL whose Tail is greater than Ring Length. Applicable for both QD and Legacy Interfaces.
21	RESERVED_DBL_TYPE	Enable detection of DBL whose {Type} field is Reserved (2'b11) which is considered malicious.
22	ILLEGAL_HEAD_DROP_DBL	Enable detection of Head Drop DBL over Comms Queue for which Head Drop is disabled.
23	LSO_OVER_COMMS_Q	Enable detection of LSO DBL over Comms Queue for which LSO is disabled.
24	ILLEGAL_VF_QNUM	Enable detection of VF-TYPE DBLQ element which is associated with illegal Q number.
25	QTAIL_GT_RING_LENGTH	Enable detection of DBL whose QDTail is greater than Ring Length. Applicable for QD Interface.

9.2.2.2.1.4 Rx Checks (GL_MDCK_RX Register)

Table 9-9 describes the checks that are done by the E810 on data fetches for Rx.

Table 9-9. Malicious Driver - Tx-Descriptor Checks (GL_MDCK_TDAT_TCLAN)

Check Type	GL_MDET_RX Bit	Event ID on MDET_RX Register	Comments
Rx-Descriptor Address	0 DESC_ADDR	1	Descriptor fetch failed[1]

1. A read can time-out or return a UR, It might possibly be poisoned or it might be internally blocked for being outside of the physical address space. The PCIe error registers can be read to distinguish between the reasons.

When a malicious event is detected in a receive queue, all packets received by this queue are dropped and no descriptors are fetched until the queue is re-initialized according to the following flow:

1. Disable queue.
2. Re-configure ring head/tail.
3. Reconfigure context (at least clear the *Queue_Block Indication* bit).
4. Enable queue.

9.2.2.3 Hardware Resources Not Assigned to VFs

Certain device capabilities are not exposed to VFs, neither directly nor via a PF. The following features are available only to PFs or controlled solely by the PF:

- Power management and WoL are PF resources and are not supported per VF.
- Link Control - The link is a shared resource and as such is controllable only by the PF. This includes PHY settings, speed and duplex settings, flow control settings, and so on. Flow control packets are sent using the station MAC Address stored in the EEPROM. The watermarks of the flow control process and the timeout value are also controllable by the PF only. In a DCB environment, the parameters of per-TC flow control and the ETS settings are also PF responsibilities.
- Configuration of Control Domains and Profiles is done solely by the PFs and is not exposed to VFs
- DCB policy and configuration of the device (either via DCBX or otherwise) is not open to VFs' control. A VF can still associate its queues with specific TCs, can request (via its PF) for Tx-Scheduler nodes to be added under specific VFs, and to set the appropriate QoS fields in its transmitted packets.
- Special Filtering Options - Save Bad Packets is a debug feature. As such, Save Bad Packets is available only to the PF. Bad packets are forwarded to a control VSI and thus should not be seen by a VF in regular operation.
- Reception of long packets is controlled separately per queue. As this impacts flow control thresholds, the PF should be made aware of the decisions of all VMs. Because of this, the setup of large send packets is centralized by the PF and each VF might request this setting.

9.2.2.4 Hardware Resources Assignment to VFs

Table 9-10 describes how the E810 shared resources are distributed between different VFs. Details for each type of resource can be found in the relevant section of this document.

Table 9-10. VF Resource Allocation

Resource	Allocation Method	Section Reference
General Resources:		
Tx-Queues and Rx-Queues	Dynamic allocation. Each VF can have a different number of queues (up to 256). Each PF allocates the queues to its VFs and their VSIs. Queue initialization is done by the PF. Following initialization, the VF manages its queues for Tx/Rx operation.	10.2 10.2.2
Admin Queues	An Admin Queue is allocated to each VF to communicate with its PF. The queue accesses a shared mailbox in the device. In SNR-BTS, a DSI VF is allocated multiple mailbox AQs for its device drivers to communicate with each other.	9.5
Interrupt Causes and Vectors	The allocation of interrupt vectors to VFs is fully flexible, loaded from NVM with a limit on the max number of vectors. Assignment of interrupt causes: • Causes for Tx-Queues, Tx-Queues, PE Queues, Admin Queues to interrupt vectors are done by each VF. • Other causes are handled only by the PFs. The PF driver can then propagate the interrupt to the VFs via Admin Queues or some other mechanism. VFs operate with MSI-X interrupts only.	9.1
Transmit Scheduling	The Scheduler Tree can be programmed to assign a subtree to a VF and its resources (for example, to its VSIs, and its queues). The nodes in the sub-tree can be configured with all supported attributes, such as guaranteed bandwidth, rate limiting, and arbitration scheme. Configuration is done by the PF and not directly by the VF.	8.3

Table 9-10. VF Resource Allocation [continued]

Resource	Allocation Method	Section Reference
Stateless Offloads	**Tx:** All the regular transmit offloads, like checksum and TSO, are available to VFs. Enabling these offloads is done by the PF as part of the queue initialization process. **Rx:** All the regular receive offloads, like checksum and header split, are available to VFs. Enabling these offloads is done by the PF as part of the queue initialization process.	10.4.4 (Rx) 10.5.8 (Tx)
Packet Analyzer		
ACLs	A VF can request its PF to add/modify/remove an ACL entry on its behalf. The entry can be potentially associated with the VSI or the VF.	7.9.2
Statistics	The VF is not directly exposed to statistics counters. If a VF needs to get statistics for its traffic, it is done via its PF.	9.6
IEEE 1588	IEEE 1588 is a per-link function and thus is controlled by the PF driver.	
RDMA Resources:		
RDMA Capabilities	32 VFs can have RDMA capabilities. The GLPE_VFPFMAP table is used to map RDMA enabled VFs to the RDMA register sets.	11.10
QPs/CQs/CQEs/SDs	According to the profile of the device, the QP/CQ/CQE/SD resources are distributed evenly between the physical functions supporting iWARP. The resources of a PF can be evenly distributed between the VFs supporting RDMA.	
CSRs	All VFs have a set of registers to control RDMA functionality. However, only 32 of these sets are active. The RDMA register set contains registers in regular VF CSR space. These registers are accessible to the VF driver and provide a common doorbell page accessible to all the userspace processes in the VF.	
Switching Resources:		
Switch Resources	Switch configuration and resource allocation are not done by a VF directly, but through its PF. See more details in the lines below.	7.8
VSIs	Up to 768 VSIs can be directly supported by the E810. These VSI resources are distributed between the different PFs dynamically. Each VF is guaranteed to receive at least one VSI. The PF can allocate multiple VSIs to a VF. There is no limit on the number of VSIs that can be assigned to a VF apart from the limit due to the number of Tx/Rx-Queue pairs the VF is allocated.	7.8.8
Unicast MAC Addresses, Multicast addresses, and VLAN tags	The PF driver is responsible for the resource allocation to its VFs. The E810 does not manage the per-VF resources of the switch. The PF assigns filters to VFs VSI using the admin commands listed in Section 7.8.12.5.2.	7.8.10
Filtering Resources:		
RSS	Directly controlled by each VF. RSS per VSI with either Small or Medium RSS Table.	7.10.7
Flow Director	Flexible population of FD entries with VF rules. Programming is done through the PF or directly by a VF.	7.10.8
Quad Hash Filtering	Each PF allocates a private memory region for itself and its VFs. Each PF populates its private memory region with its VFs' entries. The on-die Quad Hash filter caches entries for both PFs and VFs	7.10.9

9.2.3 Scalable I/O and PASID

Scalable I/O Virtualization is a way to expose thousands of lightweight Assignable Device Interfaces (ADIs) to non-Kernel processes without paying the cost of a lot of SR-IOV functions.

Each ADI exposes part of the fast path interface of the device to a process that can directly define the buffer content with no need of intervention by the regular Kernel driver for par packet processing.

Each ADI has the same trust level as a VF. The address translation of GPA to HPA uses the PASID tag that is appended to all the TLPs associated with a process instead of the BDF used in SR-IOV. The VT-d uses this information to make sure the process is only accessing pages it is allowed to and with the right access type (R/W).

All the malicious driver protections are in effect for queues assigned to VMs via a valid PASID.

9.2.3.1 Assumptions

The capabilities to be exposed as part of an ADI are:

- Tx-Queues (Quanta queues access uses the same PASID as the matching Tx-Queue)
- Rx-Queues
- RDMA QPs
- Interrupts
- A mailbox queue

All other capabilities, such as Doorbell Queues and Completion Queues, are either not used for PASID scenarios, or owned by the PF driver.

9.2.3.2 PASID

PASID is a new capability of PCIe 3.1 that allows identification of DMA traffic belonging to specific flows. It is used by the VT-d engine to do the GPA-to-HPA translation.

9.2.3.2.1 PASID Context

The PASID context is stored per VSI as described below:

- VSI_PASID.*PASID*[19:0]
- VSI_PASID.*EN*[31]

This context is used for all the LAN descriptors and data buffers transactions, and RDMA transactions associated with this VSI.

Note: VSI_PASID.*EN* should not be set for VSIs belonging to a PF that did not enable PASID in its config space.

9.2.3.2.2 PASID Stop Mechanism

The specification requires support of a mechanism to gracefully stop using a specific PASID. The mechanism to support this is disabling of all the queues associated with this PASID.

9.2.3.3 Assignable Device Interface

The following resources can be assigned to an ADI:

- **A mailbox** — Exposed through the 4K page range of mailboxes (32-35 MB).
- **A set of interrupts** — Exposed through the 4K page range of interrupts (48-56 MB).
- **A set of Rx-Queues** — Exposed through the 4K page range of Rx-Queues (56 - 64 MB).
- **A set of Tx-Queues** — Exposed through the 4K page range of Tx-Queues (64 - 128 MB).
- **MSI-X vectors** — Accessible to the PF either through regular MSI-X BAR or through a table in the regular BAR.
- **RDMA QPs** — Using regular user mode RDMA interface.

To be able to access these registers, the BAR0 userspace extension should be enabled by setting GLPCI_LBARCTL.*PAGES_SPACE_EN_PF* bit.

9.3 Host Memory Cache

The E810 uses host memory as backing store for a number of context objects used to track queue state and iWARP objects. The Host Memory Cache (HMC) is the component responsible for managing the iWARP context objects stored in host memory. The HMC manages host memory on a per PCI function basis and further breaks down each PCI function's HMC memory space into memory used to manage each context object that is in use for a given PCI function. Host software is responsible for allocation of the host pages used by the HMC before accessing a specific object. Additionally, the amount of memory that can be used for HMC backing store for a specific function is dictated by the active resource profile, which is determined by the software driver's operating environment and the number of PCI functions that are currently active. Resource profiles can be selected at driver initialization time.

9.3.1 Host Memory Usage

The HMC requires backing store for numerous data structures to be resident in host memory to perform its functions. Table 9-11 provides a list of the data structures and the amount of memory that needs to be allocated for each data structure. The "HMC Object Location" column indicates if the HMC object (and the associated backing store pages) is located only in the PF HMC object space or if it is located in both the PF and the VF HMC object space.

In general, Protocol Engine HMC objects are separated into PF- and VF-specific HMC object spaces. The resources can be sparsely populated. For example, if a function is allotted 512 QPs and only eight are used, only 4K of memory needs to be allocated, not the entire memory for all 512 QPs. Some HMC objects need to be fully populated at driver initialization, such as Protocol Engine Hash Table entries. See Section 11.5 for more information on the HMC resource allocation policies for the Protocol Engine.

Table 9-11. HMC Objects

HMC Object	HMC Object Location	Size (Bytes)	Max Quantity	Description
Protocol Engine QP Context	PE-enabled PFs and VFs	512	256K	512 bytes are reserved for QP context. There is a maximum of 256K QP contexts per device. This number is divided amongst all Protocol Engine enabled PCI functions. The memory for the VF objects is allocated by the PF driver and accessed using the PF Requester ID (RID). In the case of a single function device, hardware reserves one of the 256K QPs for internal use.
Protocol Engine CQ Context	PE-enabled PFs and VFs	64	512K	64 bytes are reserved for CQ context. There is a maximum of 512K CQ contexts per device. This number is divided amongst all Protocol Engine enabled PCI functions. The memory for the VF objects is allocated by the PF driver and accessed using the PF Requester ID (RID).
Protocol Engine TCP Timers	PE-enabled PFs and VFs	64	2^{28} per PCI function	This structure is allocated per PCI function. The number of elements allocated for this structure is dependent on the number of QPs associated with the PCI function. The equation for the number of objects is: ((ROUNDUP512(number of QPs)/512) + 1) * 4096. The memory for the VF objects is allocated by the PF driver and accessed using the PF Requester ID (RID).
Protocol Engine Hash Table Entry	PE-enabled PFs and VFs	64	(#Protocol Engine QPs + #Multicast Groups) * HTMULTIPLIER per function	This structure is allocated per PCI function. The number of elements allocated for this structure needs to be: (round_up_512(number of QPs + number of Multicast Groups) rounded up to the next power of two) * HTMULTIPLIER. The memory for the VF objects is allocated by the PF driver and accessed using the PF Requester ID (RID). The actual number of Hash Table entries for the PF is programmed using the PFQF_PE_CTL1, PFQF_PE_CTL2, VPQF_PE_CTL1, and VPQF_PE_CTL1 registers.
ARP Table Entry	PE-enabled PFs and VFs	16	65536 per PCI function	The maximum size of the ARP table is 65536 entries per Protocol Engine enabled PCI function. The memory for the VF objects is allocated by the PF driver and accessed using the PF Requester ID (RID).
Accelerated Port Bit Vector In-Use	PE-enabled PFs and VFs	8K	1 per PCI function	This table is used for the E810 to track each PCI functions usage of the Accelerated Port Bit Vector Table, and is only required for Protocol Engine enabled PCI functions. The memory for the VF objects is allocated by the PF driver and accessed using the PF Requester ID (RID).
Memory Region Table Entry (MRTE)	PE-enabled PFs and VFs	32	4M per PCI function	Each Protocol Engine enabled PCI function can have up to 4M MRTE entries allocated. The memory for the VF objects is allocated by the PF driver and accessed using the PF Requester ID (RID).
Physical Buffer List Entry (PBLE)	PE-enabled PFs and VFs	8	256M per PCI function	Each Protocol Engine enabled PCI function can have up to 256M PBLE entries allocated. The memory for the VF objects is allocated by the VF driver and accessed using the VF Requester ID (RID). Any associated Page Descriptors are accessed allocated by the PF driver and accessed using the PF RID.

Table 9-11. HMC Objects [continued]

HMC Object	HMC Object Location	Size (Bytes)	Max Quantity	Description
Xmit FIFO	PE-enabled PFs and VFs	32	128M per PCI function	Each Protocol Engine enabled PCI function can have up to 128M Xmit FIFO entries per PCI function. Xmit FIFO entries are used to track unacknowledged Protocol Engine Send Queue work requests (WQEs). The number of XMIT FIFO entries to allocate for a given PCI function can be calculated based on expected network performance and workload pattern. Allocating too few XMIT FIFO entries will result in head-of-line blocking for PE QPs running on a particular PCI function. The memory for the VF objects is allocated by the PF driver and accessed using the PF Requester ID (RID). Must be specified as a power of two.
Xmit FIFO Free List	PE-enabled PFs and VFs	4	32M per PCI function	Each Protocol Engine enabled PCI function can have up to 32M Xmit FIFO Free List entries per PCI function. The memory for the VF objects is allocated by the PF driver and accessed using the PF Requester ID (RID).
Inbound RDMA Read Queue (IRRQ or Q1)	PE-enabled PFs and VFs	64	128M per PCI function	Each Protocol Engine enabled PCI function can have up to 128M IRRQ entries. IRRQ entries are allocated on a per QP basis and in quanta's of the maximum number of outstanding RDMA Reads that are allowed multiplied by 2. In other words, if a PE-enabled PCI function has 64K QPs enabled and each is allowed up to 64 outstanding RDMA Reads, space for 8M (64K*(64*2)) IRRQ entries (512MB) must be allocated by the driver. The memory for the VF objects is allocated by the PF driver and accessed using the PF Requester ID (RID). Must be specified as a power of two.
Inbound RDMA Read Queue (IRRQ or Q1) Free List	PE-enabled PFs and VFs	4	32M per PCI function	Each Protocol Engine enabled PCI function can have up to 32M IRRQ Free List entries. The memory for the VF objects is allocated by the PF driver and accessed using the PF Requester ID (RID).
Read Response FIFO	PE-enabled PFs and VFs	32	128M per PCI function	Each Protocol Engine enabled PCI function can have up to 128M Read Response FIFO entries per PCI function. Read Response FIFO entries are used to track unacknowledged RDMA Read work requests (WQEs). The number of Read Response FIFO entries to allocate for a given PCI function can be calculated based on expected network performance and workload pattern. Allocating too few Read Response FIFO entries will result in head-of-line blocking for PE QPs running on a particular PCIe function. The memory for the VF objects is allocated by the PF driver and accessed using the PF Requester ID (RID). Must be specified as a power of two.
Read Response FIFO Free List	PE-enabled PFs and VFs	4	32M per PCI function	Each Protocol Engine enabled PCI function can have up to 32M Read Response FIFO Free List entries per PCI function. The memory of the VF objects is allocated by the PF driver and accessed using the PF Requester ID (RID).
Header	PE-enabled PFs and VFs	64	256K	64 bytes are reserved for the QP-specific static data, which is used when building a packet. There is a maximum of 256K Header objects per device (one per QP). This number is divided amongst all Protocol Engine enabled PCI functions. The memory for the VF objects is allocated by the PF driver and accessed using the PF Requester ID (RID).

Table 9-11. HMC Objects [continued]

HMC Object	HMC Object Location	Size (Bytes)	Max Quantity	Description
Metadata	PE-enabled PFs and VFs	16	8M	128 bytes are reserved for metadata, which is used when building a packet. It is the packet specific, non-static data. There are a maximum of 1M Metadata objects per device. This number is divided amongst all Protocol Engine enabled PCI functions. The memory for the VF objects is allocated by the PF driver and accessed using the PF Requester ID (RID).
Out-of-Order Send Completion (OOISC) FIFO	PE-enabled PFs and VFs	32	256K	32 bytes are reserved for the Out-of-Order Send Completion Object. There is one of these per QP. It is the packet specific, non-static data. This number is divided amongst all Protocol Engine enabled PCI functions. The memory for the VF objects is allocated by the PF driver and accessed using the PF Requester ID (RID).
Out-of-Order Send Completion (OOISC) FIFO Free List	PE-enabled PFs and VFs	4	256K	Each Protocol Engine enabled PCI function can have up to 256K OOISC FIFO Free List entries. The memory for the VF objects is allocated by the PF driver and accessed using the PF Requester ID (RID).
Multicast Group	PE-enabled PFs	64	8192K per PCI function	Each Protocol Engine enabled PCI Physical Function can have up to 8K multicast groups. Virtual Functions do not have multicast groups. The memory for the VF objects is allocated by the PF driver and accessed using the PF Requester ID (RID).
Address Handles	PE-enabled PFs and VFs	64	128K per PCI function	Each Protocol Engine enabled PCI Function can have up to 128K Address Handles. The memory for the VF objects is allocated by the PF driver and accessed using the PF Requester ID (RID).

To access (and cache in on-chip memory) the data structures defined in Table 9-11, the HMC uses the concept of private memory address space. The E810 has an 8GB private memory address space that can be sparsely backed with host memory based on actual context usage. Drivers do not need to allocate pages for HMC objects that are not currently being used by the driver. The private memory address space is first broken down by PCI function, then by object or data structure type, and finally by object index. The portion of the private memory address space that is allocated to a particular PCI function is termed Function Private Memory (FPM). Also note that the VF FPMs are not programmed directly by the VF drivers. The PF driver uses the HMC function index to select the VF FPM to be programmed.

Figure 9-5 shows how the E810 provides the address mapping between Private Memory and Host Physical Addresses. PM address shown on the left side of the figure indicates E810 Private Memory address from 0 to 8GB-1. The E810 works with PM address space internally, which is converted to Host Physical Addresses in order to access host memory.

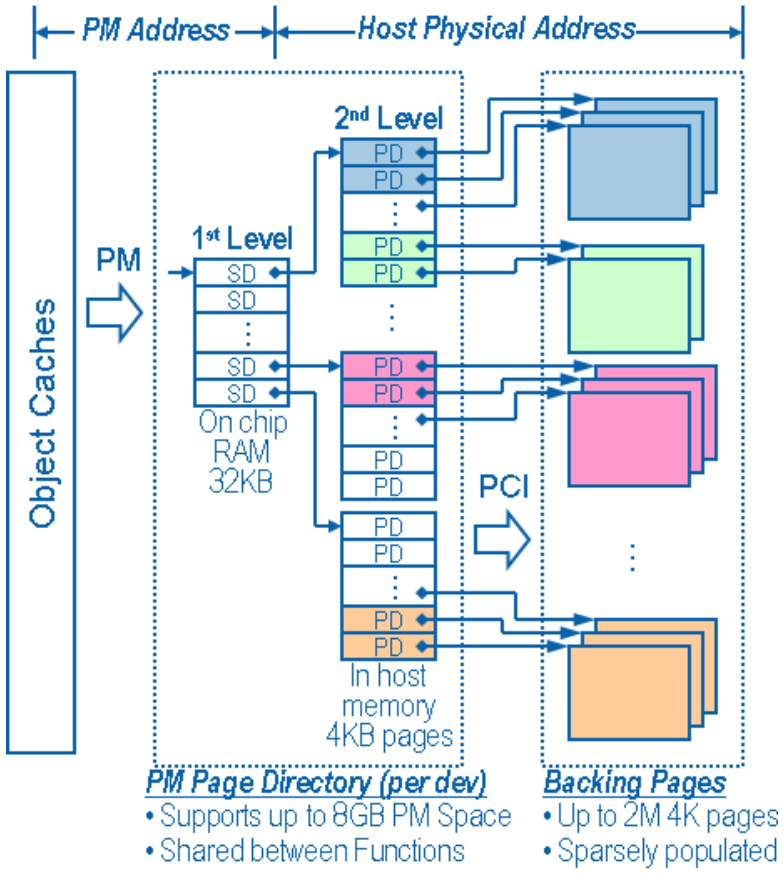


Figure 9-5. Host Memory Cache Private Memory Address Space

See Section 9.3.2 for information on the Quad Hash cache.

The left portion of Figure 9-5 shows portions of the HMC that are resident on-chip. This portion includes the actual object caches that retain portions of the data from host memory to improve performance and the Segment Descriptors (SD). The SDs reside in a 32 KB RAM on-chip called the Segment Descriptor Table. The Segment Descriptor Table holds 4096 pointers to host memory pages. Unique ranges of sequential SDs in the Segment Descriptor Table are allocated to a each PCI function that is active. The Segment Descriptor Table is the first level of private memory address translation provided by the E810. SDs are programmed using the PFHMC_SDCMD (Section 13.2.2.20.35), PFHMC_SDDATALOW (Section 13.2.2.20.36), and PFHMC_SDDATAHIGH (Section 13.2.2.20.37) registers. Protocol Engine CQP operations can also be used to program Segment Descriptors.

Everything to the right of the Segment Descriptor Table in Figure 9-5 resides in host memory. Each PCI function has a set of registers (GLHMC_SDPART[n] and GLHMC_VFSDPART[n]) that define the base and number of SDs that belong to the PCI function. The GLHMC_SDPART[n] and GLHMC_VFSDPART[n] registers are programmed from NVM and can also be programmed by firmware during the Create Control Queue Pair operation (Section 11.5.2.1).

The E810 provides range checking for each internal access to ensure that a given PCI function is never allowed to access memory outside of its valid range of SDs. The E810 manages the SD base and number registers internally based on the resource profile that is either loaded at NVM load or selected by the first E810 driver to load for a device during the Create Control Queue Pair operation (Section 11.5.2.1).

The second level of private memory address translation provided by the E810 is Page Descriptors (PDs). Each SD points to a single host page that is divided into 512 PDs that are simply 64-bit physical memory addresses. Each PD points to a backing page for the private memory address space. The total 8 GB private memory address space is derived using a fully-populated Segment Descriptor Table pointing to 4096 4KB Host pages that hold the 2M PDs. Each of the 2M PDs point to host memory backing pages for a total of 8 GB of address space. As previously mentioned, there is no requirement to populate all SDs or PDs with memory if the portion of private memory address space is not in use by software. The format of the PD structure in host memory is shown in Table 9-12.

Table 9-12. HMC Page Descriptor Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:12]	HMC Backing Page Physical Address	[0]	PD_Valid
	[11:1]	Reserved		

The HMC Backing Page Physical Address is the address of a driver allocated page that will hold HMC object context. This address must be aligned to a 4 KB address in host memory. The PD Valid bit allows software to sparsely-populate the PD entries on an as-needed basis once HMC context objects are needed. Software must allocate host memory pages which hold packed arrays of PDs as shown in Figure 9-5. The physical addresses of these PD pages are used to populate SD entries by using the PFHMC_SDCMD (Section 13.2.2.20.35), PFHMC_SDDATALOW (Section 13.2.2.20.36), and PFHMC_SDDATAHIGH (Section 13.2.2.20.37) registers. For more information on how to program SDs using these registers, see Section 9.3.8.

The Private Memory is further divided into separate PCI Function Private Memory (FPM) Addresses. A PCI function can be either a physical function or a virtual function. The first eight FPM address spaces are reserved for NIC PFs. The next eight FPM address spaces are used for PFs that support Protocol Engine provided accelerations for iWARP, RoCEv2, or UDA. The final 32 FPM address spaces are used for VFs that support Protocol Engine provided accelerations for iWARP, RoCEv2, or UDA.

Figure 9-6 shows how the private memory address space is divided up for each PCI function. The smallest amount of private memory that can be allocated to a function is 2 MB (1 SD). The maximum that could be allocated to a function would be the entire segment table, in which case no other function can have any private memory resources. Note that the object caches address HMC objects using HMC function number to determine the correct FPM. The FPM identifies the range of private memory address space that belongs to a PCI function. Since each SD represents 2 MB of HMC PM address space, the FPM also identifies the range of SDs that belong to a PCI function.

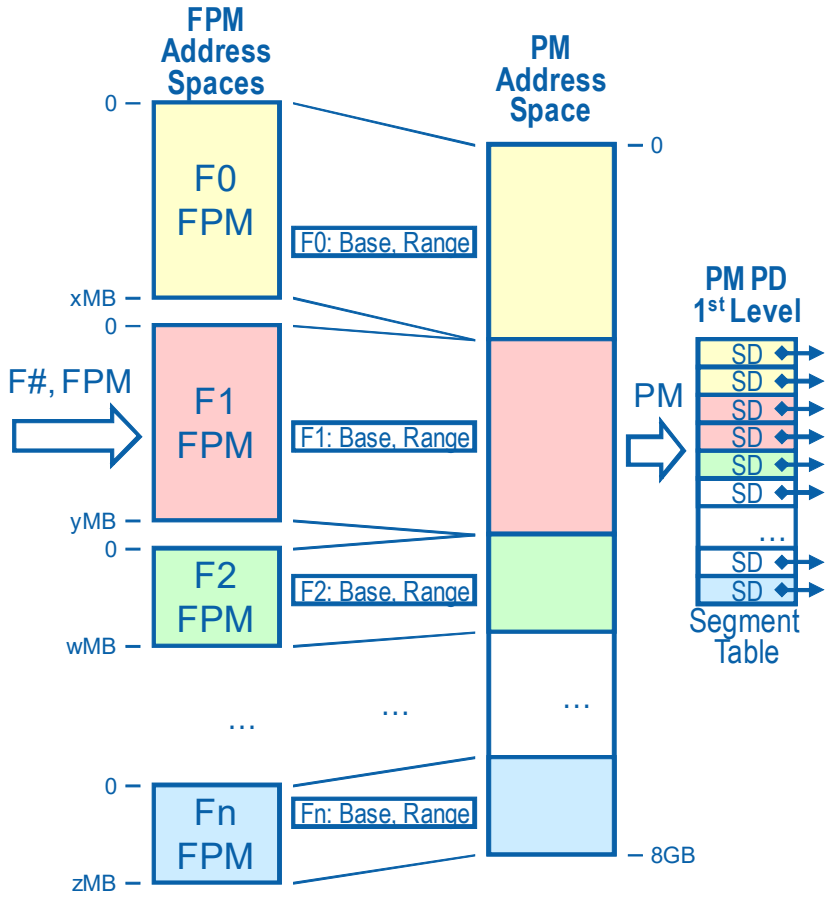


Figure 9-6. Host Memory Cache Function Private Memory Space

Each PCI Function's Private Memory space is further divided into separate memory spaces for each object in host memory. Each PCI function has a set of registers per function that define the object's base address in FPM space and the bounds (or maximum number of entries) of a particular object.

Figure 9-7 depicts some of the current objects that reside in the private memory space (see Table 9-11 for a complete list of the objects). The FPM address is calculated based on the object type (which identifies the object base register) and object index (the FPM base has already been calculated). Ultimately, the FPM base address, object base address, object size, and object index are all used to determine the private memory address.

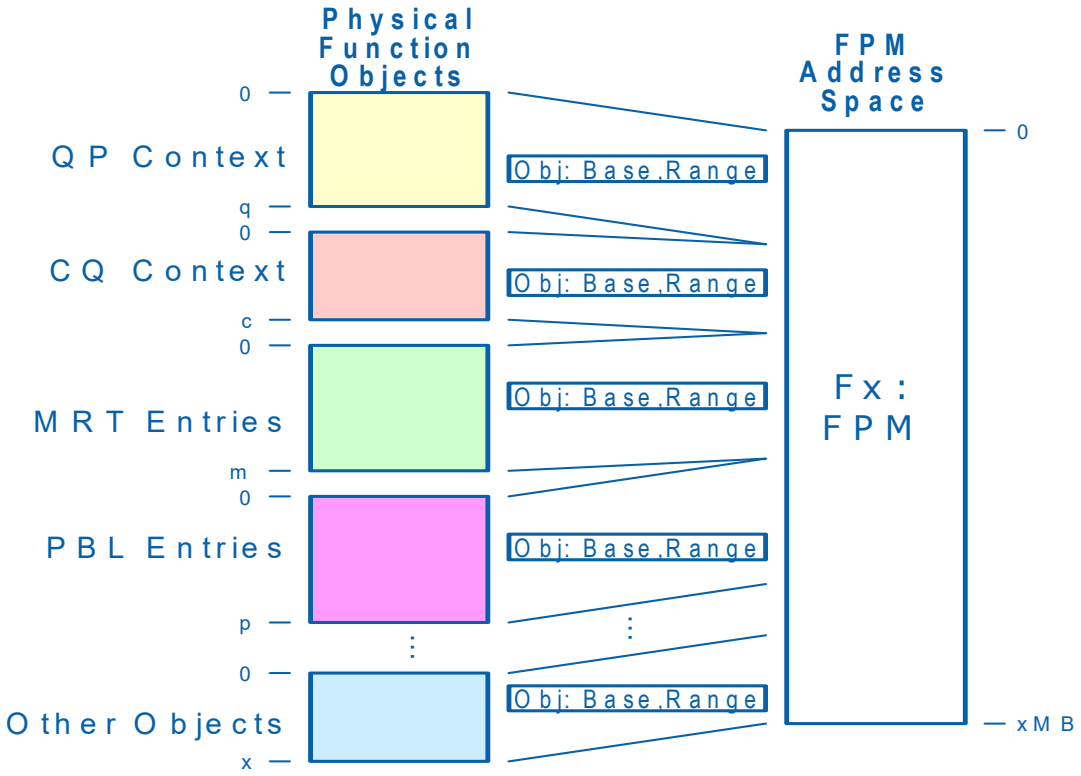


Figure 9-7. Host Memory Cache FPM Object Access

Figure 9-8 describes the decoding of the Private Memory Address into Host Address. Additionally, Figure 9-8 depicts an SD addressing a private memory space backing page directly instead of using the second level of indirect addressing (PD).

Each PCI function can set any SD within its range of SDs to be either pointing to a PD or directly to a backing page. The segment type is specified in the PFHMC_SDDATALOW.*PMSDTYPE* register field. The direct segment approach can be used for PCI functions that do not have large requirement for FPM space to reduce overhead incurred while accessing HMC objects.

Additional usage of the direct segment approach is possible if the driver is able to allocate a physically-contiguous range of pages large enough to hold the entire PD space needed to support the FPM required by the driver loading on a specific PCI function. This mode prevents an additional address lookup and increases the performance if the driver happens to allocate a block of physically-contiguous memory, or the operating system has support for 2 MB pages.

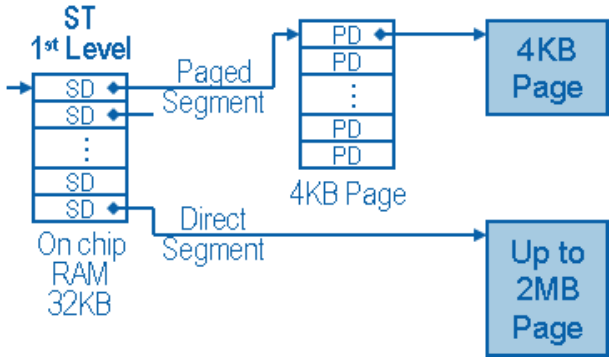


Figure 9-8. Host Memory Cache Direct Segment

See Section 9.3.7 for more details on the specific formats of the SD entries for Paged and Direct addressing modes.

9.3.2 Object Caches

The Host Memory Cache for RDMA is split into two caches. One cache contains all RDMA objects except the Quad Hash entries. The Quad Hash entries are placed in a different cache so they can be referenced by the filtering mechanism.

When Quad Hash objects are added or deleted, the RDMA firmware is responsible for updating the Quad Hash objects in the Quad Hash cache.

The two caches have separate registers, but share the same basic mechanisms. Since there are two components referencing the Quad Hash cache, the registers for both caches must remain consistent.

For more information on the Quad Hash Host Memory Cache (including registers and interrupts) see section Section 9.4.

9.3.3 Private Memory Space Profiles

The E810 private memory address space configuration is a two step process:

1. Partition the HMC-related resources into per PCI function resources.
2. Divide the resulting Function Private Memory (FPM) into individual objects.

To simplify resource allocation, the E810 provides a resource profile concept that takes into account the number of PCI functions, the number of Protocol Engine enabled PCI functions, and the Operating System environment to divide HMC private memory space and Protocol Engine Doorbell resources. These HMC Resource Profiles are used to partition the HMC Segment Descriptor table and Protocol Engine Doorbell resources. Software performs the FPM division on a per PCI function basis. The currently-defined HMC Resource profiles are described in Table 9-13.

Protocol Engine Doorbell resources place constraints on the HMC resource profiles since there is a fixed number of these resources on-die. There are enough doorbell resources for 256K Protocol Queue Pairs and 512K Protocol Engine Completion Queues. These resources must be partitioned between PCI functions that require Protocol Engine functionality. Protocol Engine resources are further discussed in Section 11.1.

The resource profiles drive the values found in the following registers:

- **GLHMC_SDPART** — Configures the HMC Segment Descriptor range per PF.
- **GLHMC_PFPESDPART** — Configures the Protocol Engine HMC Segment Descriptor range per PF.
- **GLHMC_VFSDPART** — Configures the HMC Segment Descriptor range per Protocol Engine enabled VF.
- **GLHMC_DBQPPART** — Configures the range of Protocol Engine QP doorbells per PF.
- **GLHMC_VFDBQPPART** — Configures the range of Protocol Engine QP doorbells per Protocol Engine enabled VF.
- **GLHMC_DBCQPART** — Configures the range of Protocol Engine CQ doorbells per PF.
- **GLHMC_VFDBCQPART** — Configures the range of Protocol Engine CQ doorbells per Protocol Engine enabled VF.
- **GLHMC_CEQPART** — Configures the range of Protocol Engine CEQs per PF.
- **GLHMC_VFCEQPART** — Configures the range of Protocol Engine CEQs per Protocol Engine enabled VF.

These registers are loaded from NVM to match the Default profile. The Protocol Engine firmware can be used to change the programming of these registers to match any of the following resource profiles.

Table 9-13. E810 HMC Resource Profiles

Profile Name	Description
Default	The default profile evenly distributes all HMC Segment Descriptor table entries and Protocol Engine doorbell resources among all active Physical Functions. No Protocol Engine resources are allocated to Virtual Functions. See Section 9.3.5 for more details.
SR-IOV VF Primary	The SR-IOV VF Primary HMC resource profile distributes a relatively small number of HMC Protocol Engine resources to the active PFs, and then evenly distributes the remaining HMC Protocol Engine resources among the Protocol Engine enabled VFs. This resource profile assumes that the Protocol Engine functionality will primarily be used by VFs when running a Virtualized Operating System. See Section 9.3.5.1 for more details.
SR-IOV Even Distribution	The SR-IOV Even Distribution HMC resource profile evenly distributes the HMC Protocol Engine resources among the PFs and Protocol Engine enabled VFs. This resource profile assumes that the Protocol Engine functionality will be used by PFs and VFs when running a Virtualized Operating System. See Section 9.3.5.2 for more details.

Each of these profiles are simply a set of equations that the E810 uses to configure the HMC. This first phase of the HMC initialization is activated by an NVM reload (all settings revert back to the default profile), or by selecting a resource profile when creating a Protocol Engine Control Queue Pair, as specified in Section 11.5.2.1.

Host software will only need to change from the default profile based upon the need for Protocol Engine functionality in a VF. If this is not the case, the default profile covers all usage models. In the case where the Protocol Engine functionality is required in a VF, the driver determines which of the two resource profiles best fits the users needs and selects that profile at driver initialization time.

9.3.4 Host HMC Resource Partitioning

The HMC resource profile can be selected by Expansion ROM code before the operating system loads, and also by the first Protocol Engine driver that loads within the operating system. It is expected that Expansion ROM boot code (PXE, iSCSI) does not need to change the profile, but it can change the profile if needed. Once the Protocol Engine operating system driver loads, it is also allowed to change the HMC resource profile.

9.3.5 Default HMC Profile Equations

The default HMC resource profile distributes all Protocol Engine resources evenly among all active PFs (no Protocol Engine Resources are available for VFs). This includes partitioning the SD table, the Protocol Engine QP and CQ doorbell resources, and the Protocol Engine Completion Event Queues. The number of active Physical Functions is reported in the GLGEN_PCIFCNCNT registers.

Table 9-14. Default HMC Profile Examples

Number of Active PFs	PF Index	SD Base Index	SD Count	PE QP Doorbell Base Index	PE QP Doorbell Count	PE CQ Doorbell Base Index	PE CQ Doorbell Count	PE CEQ Base Index	PE CEQ Count
1	0	Prot Eng: 0	Prot Eng: 4096	0	256K	0	512K	0	768
	1-7	Prot Eng: 0	Prot Eng: 0	0	0	0	0	0	0
2	0	Prot Eng: 0	Prot Eng: 2048	0	128K	0	256K	0	384
	1	Prot Eng: 2048	Prot Eng: 2048	128K	128K	256K	256K	128	384
	2-7	Prot Eng: 0	Prot Eng: 0	0	0	0	0	0	0
4	0	Prot Eng: 0	Prot Eng: 1024	0	64K	0	128K	0	192
	1	Prot Eng: 1024	Prot Eng: 1024	64K	64K	128K	128K	64	192
	2	Prot Eng: 2048	Prot Eng: 1024	128K	64K	256K	128K	128	192
	3	Prot Eng: 3072	Prot Eng: 1024	192K	64K	384K	128K	192	192
	4-7	Prot Eng: 0	Prot Eng: 0	0	0	0	0	0	0
8	0	Prot Eng: 0	Prot Eng: 512	0	32K	0	64K	0	96
	1	Prot Eng: 512	Prot Eng: 512	32K	32K	64K	64K	32	96
	2	Prot Eng: 1024	Prot Eng: 512	64K	32K	128K	64K	64	96
	3	Prot Eng: 1536	Prot Eng: 512	96K	32K	192K	64K	96	96
	4	Prot Eng: 2048	Prot Eng: 512	128K	32K	256K	64K	128	96
	5	Prot Eng: 2560	Prot Eng: 512	160K	32K	320K	64K	160	96
	6	Prot Eng: 3072	Prot Eng: 512	192K	32K	384K	64K	192	96
	7	Prot Eng: 3584	Prot Eng: 512	224K	32K	448K	64K	224	96

The calculation used by the E810 for the number of Segment Descriptors for the Protocol Engine should be SDs (4096) divided by the number of PFs, then rounded down to the next integer. The calculation for the number of PE QPs is the number of PE QPs (256K) divided by the number of PFs, then rounded down to the next full page size that would be needed for context. PE QP context size is 512 bytes per QP.

The calculation for the number of PE CQs is similar, except that the max number of PE CQs is 512K and the CQ context size is 64 bytes.

The PE CEQ calculation is similar to the Segment Descriptor Calculation, except that the number of PE CEQs total is 256 and the maximum number or CEQs for any single PCI function is 768.

Table 9-15 shows the resource counts for all possible PF counts in the Default HMC resource profile assuming that VFs are evenly distributed to active PFs.

Table 9-15. Initial Host Memory Cache Partitioning per PF

PE Count	#PE SDs	#PE QPs	#PE CQs	#PE CEQs
1	4096	262144	524288	786
2	2048	131072	256144	384
3	1365	87376	174762	256
4	1024	65536	131072	192
5	819	52424	104857	153
6	682	43688	87381	128
7	585	37448	74898	109
8	512	32768	65536	96

9.3.5.1 SR-IOV VF Primary HMC Profile Equations

The SR-IOV VF Primary HMC resource profile allocates Protocol Engine resources in favor of the VFs. In this case, the FPM spaces are broken down into two sets:

1. PFs
2. Protocol Engine Enabled VFs

The first eight FPMs are always reserved for PFs even if less than eight PFs are enabled. In the case where less than eight PFs are enabled, some of the FPMs will have zero resources allocated to them.

The next set of FPMs (8-39) are the Protocol Engine Enabled VFs. The maximum number of VFs in this set is 32, but can be limited to less by setting PE Enabled VF Count during the Create Control Queue Pair operation (Section 11.5.2.1).

In this resource profile, all enabled PFs are allocated the following resources:

- 10 SDs for PF Protocol Engine
- 1024 Protocol Engine Queue Pairs
- 2048 Protocol Engine Completion Queues
- 8 Protocol Engine Completion Event Queues

After allocating the resources for the PFs and the NIC VF resources that reside in the PF, the remaining resources are evenly distributed across all Protocol Engine enabled VFs. Assuming that there are two PFs active and 128 VFs (32 of which can be enabled for Protocol Engine support), the HMC resources would be distributed as shown in Table 9-16.

Table 9-16. Example SR-IOV VF Primary Profile Resource Partitioning per FPM

Function Type	FPM Index Range	#SDs	#PE QPs	#PE CQs	#PE CEQs
PF (2)	Prot Eng: 8-9	Prot Eng: 10	1024	2048	8
Inactive PF (6)	Prot Eng: 10-15	Prot Eng: 0	0	0	0
Protocol Engine Enabled VFs (32)	Prot Eng: 16-47	Prot Eng: 127	8128	16256	23
Totals		**4084**	**262144**	**524288**	**752**

The equations for distributing PE QPs and CQs are similar to those used to distribute resources among PFs in the default HMC resource profile.

9.3.5.2 SR-IOV Even Distribution HMC Profile Equations

The SR-IOV Even Distribution HMC profile is used for environments where there is equal need for Protocol Engine resources in the PFs and all Protocol Engine Enabled VFs in operating systems that support I/O virtualization. Typically the number of Protocol Engine enabled VFs is reduced in this profile to allow sufficient Protocol Engine resources for a VF to run HPC workloads.

In this resource profile, all enabled PFs are allocated the following resources:

- 1 SDs for PF NIC
- Remaining SDs/(Active PFs+RDMA Enabled VFs) SDs for Protocol Engine
- 256K/(Active PFs+RDMA Enabled VFs) Protocol Engine Queue Pairs
- 512K/(Active PFs+RDMA Enabled VFs) Protocol Engine Completion Queues
- 256/(Active PFs+RDMA Enabled VFs) Protocol Engine Completion Event Queues

In this resource profile, all enabled VFs are allocated the following resources:

- Remaining SDs/(Active PFs+RDMA Enabled VFs) SDs for Protocol Engine
- 256K/(Active PFs+RDMA Enabled VFs) Protocol Engine Queue Pairs
- 512K/(Active PFs+RDMA Enabled VFs) Protocol Engine Completion Queues
- 256/(Active PFs+RDMA Enabled VFs) Protocol Engine Completion Event Queues

Assuming the two PFs are enabled, 128 VFs are enabled, and 16 Protocol Engine enabled VFs are configured, Table 9-17 shows the distribution of resources to the HMC FPMs.

Table 9-17. SR-IOV Even Distribution Profile Resource Partitioning per FPM

Function Type	FPM Index Range	#SDs	#PE QPs	#PE CQs	#PE CEQs
PF (2)	Prot Eng: 8-9	Prot Eng: 22	14560	29127	42
Inactive PF (14)	Prot Eng: 10-15	Prot Eng: 0	0	0	0
Protocol Engine Enabled VFs (16)	Prot Eng: 16-31	227	14560	29127	42
Totals		**4088**	**262080**	**524286**	**756**

9.3.6 Function Private Memory Space

Once NVM has set the default profile or the profile was changed during the Create Control Queue Pair operation (Section 11.5.2.1), the driver can then continue on with the second step of HMC configuration, which is to break down the FPM space into individual object regions. To do this, the driver must perform the following steps:

1. Determine the HMC function index to be configured. In the case of a PF, the HMC function number is equal to the PCI function number. In the case of a VF when the Protocol Engine resources are configured, the HMC index depends on the programming of the HMC PM Function table using the CQP Manage HMC PM Function Table operation. See Section 11.5.3.3 for more information on the CQP operation.

   Note: There are two arrays defined for each HMC FPM register. One array for PFs and a second array for the Protocol Engine enabled VFs.

2. For Protocol Engine objects, read the current HMC configuration information to determine how to calculate the numbers of each HMC object that needs to be requested.

   a. The GLHMC_SDPART[n] (or GLHMC_VFSDPART[n] in the case of a Protocol Engine enabled VF) register associated with the HMC FPM reports the currently available SDs for a given PCI function, which can be used to calculate the maximum FPM space.

   b. GLHMC_DBQPPART[n] (or GLHMC_VFDBQPPART[n] in the case of a Protocol Engine enabled VF) reports the maximum number of Protocol Engine QPs that can be used. The driver does not have to allocate maximum FPM space based on PMDBMAXQP if less QPs are needed for the active driver deployment model.

   c. GLHMC_DBCQPART[n] (or GLHMC_VFDBCQPART[n] in the case of a Protocol Engine enabled VF) is used to determine the maximum number of Protocol Engine Completion Queues that can be used by a given PCI function. Similarly to the maximum QP count, the driver is not required to allocate the maximum amount of FPM space for CQs if the driver deployment model requires less.

   d. GLHMC_CEQPART[n] (or GLHMC_VFCEQPART[n] in the case of a Protocol Engine enabled VF) is used to determine the maximum number of Protocol Completion Event Queues that can be used with a given PCI function.

      Note: Protocol Engine CEQs do not consume FPM space, but are still partitioned as part of the HMC resource profiles.

   e. Other object maximum values are found in the registers named GLHMC_{object}MAX. See Table 9-18 for the specific register names.

3. Each FPM object size register is written with the minimum of the values determined in Step 2 or the actual driver needs.

For Protocol Engine objects, issue the Commit FPM Values CQP operations to program the object base registers.

Table 9-18 describes all the HMC objects and the registers used to determine the object location within Function Private Memory space, the size, and limit of each object.

Table 9-18. FPM Object Registers

HMC Object	Base Register Array	Object Counter Register Array	Maximum Object Count Register	Object Element Size Register
Protocol Engine QP Context	GLHMC_PEQPBASE GLHMC_VFPEQPBASE	GLHMC_PEQPCNT GLHMC_VFPEQPCNT	GLHMC_DBQPMAX	GLHMC_PEQPOBJSZ
Protocol Engine CQ Context	GLHMC_PECQBASE GLHMC_VFPECQBASE	GLHMC_PECQCNT GLHMC_VFPECQCNT	GLHMC_DBCQMAX	GLHMC_PECQOBJSZ
Protocol Engine TCP Timers	GLHMC_PETIMERBASE GLHMC_VFPETIMERBASE	GLHMC_PETIMERCNT GLHMC_VFPETIMERCNT	GLHMC_PETIMERMAX	GLHMC_PETIMEROBJSZ
Protocol Engine Hash Table Entry	GLHMC_PEHTEBASE GLHMC_VFPEHTEBASE	GLHMC_PEHTCNT[1] GLHMC_VFPEHTCNT	GLHMC_PEHTMAX	GLHMC_PEHTEOBJSZ
ARP Table Entry	GLHMC_PEARPBASE GLHMC_VFPEARPBASE	GLHMC_PEARPCNT GLHMC_VFPEARPCNT	GLHMC_PEARPMAX	GLHMC_PEARPOBJSZ
Accelerated Port Bit Vector In-Use	GLHMC_APBVTINUSEBASE GLHMC_VFAPBVTINUSEBASE	N/A (1 table per function)	N/A (1 table per function)	N/A (8KB fixed)
Memory Region Table Entry (MRTE)	GLHMC_PEMRBASE GLHMC_VFPEMRBASE	GLHMC_PEMRCNT GLHMC_VFPEMRCNT	GLHMC_PEMRMAX	GLHMC_PEMROBJSZ
Xmit FIFO	GLHMC_PEXFBASE GLHMC_VFPEXFBASE	GLHMC_PEXFCNT GLHMC_VFPEXFCNT	GLHMC_PEXFMAX	GLHMC_PEXFOBJSZ
Xmit FIFO Free List	GLHMC_PEXFFLBASE GLHMC_VFPEXFFLBASE	GLHMC_PEXFFLCNT GLHMC_VFPEXFFLCNT	GLHMC_PEXFFLMAX	N/A (4B fixed)
Inbound RDMA Read Queue (IRRQ or Q1)	GLHMC_PEQ1BASE GLHMC_VFPEQ1BASE	GLHMC_PEQ1CNT GLHMC_VFPEQ1CNT	GLHMC_PEQ1MAX	GLHMC_PEQ1OBJSZ
Inbound RDMA Read Queue (IRRQ or Q1) Free List	GLHMC_PEQ1FLBASE GLHMC_VFPEQ1FLBASE	GLHMC_PEQ1FLCNT GLHMC_VFPEQ1FLCNT	GLHMC_PEQ1FLMAX	N/A (4B fixed)
Multicast Group	GLHMC_FSIMCBASE GLHMC_VFFSIMCBASE	GLHMC_FSIMCCNT GLHMC_VFFSIMCCNT	GLHMC_FSIMCMAX	GLHMC_FSIMCOBJSZ
Address Handles	GLHMC_FSIAVBASE GLHMC_VFFSIAVBASE	GLHMC_FSIAVCNT GLHMC_VFFSIAVCNT	GLHMC_FSIAVMAX	GLHMC_FSIAVOBJSZ
Physical Buffer List Entry (PBLE)	GLHMC_PEPBLBASE GLHMC_VFPEPBLBASE	GLHMC_PEPBLCNT GLHMC_VFPEPBLCNT	GLHMC_PEPBLMAX	N/A (8B fixed)
Read Response FIFO	GLHMC_PERRFBASE GLHMC_VFPERRFBASE	GLHMC_PERRFCNT GLHMC_VFPERRFCNT	GLHMC_PERRFMAX	GLHMC_PERRFOBJSZ
Read Response FIFO Free List	GLHMC_PERRFFLBASE GLHMC_VFPERRFFLBASE	GLHMC_PERRFFLCNT GLHMC_VFPERRFFLCNT	GLHMC_PERRFFLMAX	N/A (4B fixed)
Header	GLHMC_PEHDRBASE GLHMC_VFPEHDRBASE	GLHMC_PEHDRCNT GLHMC_VFPEHDRCNT	GLHMC_PEHDRMAX	GLHMC_PEHDROBJSZ
Metadata	GLHMC_PEMDBASE GLHMC_VFPEMDBASE	GLHMC_PEMDCNT GLHMC_VFPEMDCNT	GLHMC_PEMDMAX	GLHMC_PEMDOBJSZ
Out-of-Order Send Completion (OOISC) FIFO	GLHMC_PEOOISCBASE GLHMC_VFPEOOISCBASE	GLHMC_PEOOISCCNT GLHMC_VFPEOOISCCNT	GLHMC_PEOOISCMAX	GLHMC_PEOOISCOBJSZ
Out-of-Order Send Completion (OOISC) FIFO Free List	GLHMC_PEOOISCFFLBASE GLHMC_VFPEOOISCFFLBASE	GLHMC_PEOOISCFFLCNT GLHMC_VFPEOOISCFFLCNT	GLHMC_PEOOISCFFLMAX	N/A (4B fixed)

1. For PFs, the sum of PFQF_PE_CTL1.PEHSIZE and PFQF_PE_CTL2.PEDSIZE must be smaller than or equal to GLHMC_PEHTCNT. For VFs, the sum of VPQF_PE_CTL1.PEHSIZE and VPQF_PE_CTL2.PEDSIZE must be smaller than GLHMC_VFPEHTCNT.

9.3.6.1 Programming the HMC FPM Base Registers

All settings of FPM registers impact only the function associated with the registers. In other words, the driver on a given PCI function must program only the GLHMC_{object}CNT and GLHMC_{object}BASE registers for HMC PMs that are owned by that same PCI function or HMC PMs that are associated with Protocol Engine enabled VFs that the PF owns. The FPM base of the first HMC object for each PCI function is always 0. The FPM base of subsequent HMC objects increment from previous HMC object base, the number of elements for the previous HMC object, and the size of the previous HMC object element. Additional rounding is necessary to get to the next FPM address that is properly aligned for the HMC object under consideration.

Table 9-19 shows the FPM object order that must be maintained for proper HMC operation and the alignment requirements for each object.

Table 9-19. FPM Object Order and Alignment

HMC Object Order	HMC Object	Alignment Requirement
1	RSVD	
2	Protocol Engine QP Context	2MB (Must be start of an SD)
3	Protocol Engine CQ Context	512B
4	Protocol Engine Hash Table Entry	512B
5	ARP Table Entry	512B
6	Accelerated Port Bit Vector In-Use	512B
7	Memory Region Table Entry (MRTE)	512B
8	Xmit FIFO	512B
9	Xmit FIFO Free List	512B
10	Inbound RDMA Read Queue (IRRQ or Q1)	512B
11	Inbound RDMA Read Queue (IRRQ or Q1) Free List	512B
12	Protocol Engine TCP Timers	512B
13	Physical Buffer List Entry (PBLE)	512B (4KB for VFs, handled by CQP firmware)
14	Multicast Group	512B
15	Address Handles	512B
16	Read Response FIFO Free List	512B
17	Read Response FIFO	512B
20	Header Information	512B
21	Metadata	512B
22	Out-of-Order Send Completion (OOISC) FIFO	512B
23	Out-of-Order Send Completion (OOISC) FIFO Free List	512B

The special case of Protocol Engine QPs must be rounded up to the next SD boundary. The base registers for Protocol Engine Objects (indexes 4 and up) can be set using the Commit FPM Values CQP operation.

Note: The driver can choose to set the GLHMC_{obj}CNT register to 0 if it does not need to utilize an object.

9.3.7 Populating HMC Backing Pages

Once the HMC resource profile has been picked (Section 9.3.3) and the Function Private Memory space has been programmed (Section 9.3.6), the driver must populate the HMC backing pages for the PCI function that it is initializing. The first step in this phase of initialization is to allocate 4KB pages for the PDs. Each 4KB PD page holds 512 PDs and occupies a single SD entry. Once a 4KB page has been allocated, initialized to zero, and pinned by software, the PFHMC_SDCMD (Section 13.2.2.20.35), PFHMC_SDDATALOW (Section 13.2.2.20.36), and PFHMC_SDDATAHIGH (Section 13.2.2.20.37) registers are used to populate the SD table for the PCI Function. Protocol Engine enabled VFs use the GLHMC_VFSDCMD[n], GLHMC_VFSDDATALOW[n], and GLHMC_SDDATAHIGH[n] registers, where n is the HMC Virtual Function index. Protocol Engine related SDs are programmed using the Update Protocol Engine SDs CQP operation.

A driver on a given PCI function mus manipulate only SD table entries that are allocated for that PCI function via the SD partitioning process that involved picking an HMC resource profile. SDs are addressed on a per PCI function basis starting at 0, and is limited by GLHMC_SDPART[n].*PMSDSIZE*. In other words, software is not aware of the actual portion of the SD table that it is using. Accesses outside of the SD range configured by NVM or a HMC resource profile using the PFHMC_SDCMD or GLHMC_VFSDCMD registers is ignored (operation is not performed) by the E810, and an error is returned in the completion for the operation that is in error.

The second step in this phase of driver initialization is to allocate additional host pages for backing HMC FPM objects for use by the E810 before the driver attempts to access the object. The breakdown of the FPM address into components is shown in Table 9-19.

Direct-mapped pages in the VF object cache must be fully populated for the entire 2MB space of the SD.

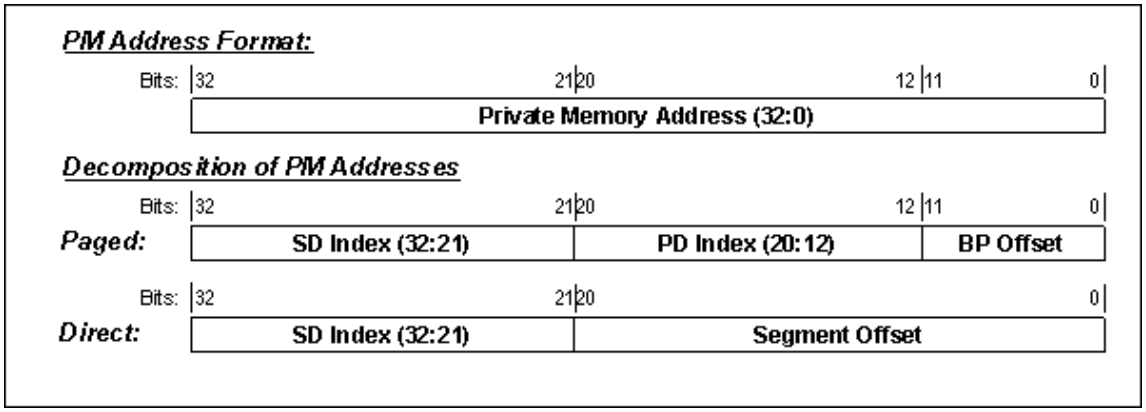


Figure 9-9. FPM Address Decomposition

The identification of which SDs to populate and which HMC FPM backing pages to populate in the PD pages can be calculated as follows in the paged scenario:

FPM_object_address = (GLHMC_{object}BASE * 512) + ($2^{GLHMC\_\{object\}OBJSZ}$ * element_index)

SD_index = INT(FPM_object_address / 2MB)

PD_index = INT(FPM_object_address / 4KB) & 0x1FF

HMC_PM_index = PF index or the HMC VF FPM index

Following is an example of populating the backing pages the HMC, assuming that a driver wants to allocate FPM backing pages for 1024 Protocol Engine QPs starting at index 8 assuming no previous SDs for Protocol Engine QPs had been allocated:

1. Allocate one PD page (capable of holding 512 backing pages of 4KB each).
2. Identify the first SD necessary:
   a. FPM_object_address = (GLHMC_PEQPBASE[HMC_PM_index] * 512) + ($2^{GLHMC\_PEQPOBJSZ}$ * 8)
   b. FPM_object_limit = FPM_object_address + ($2^{GLHMC\_PEQPOBJSZ}$ * 1024)
   c. SD_index = FPM_object_address / 2MB
   d. Last_SD_index = (FPM_object_limit - 1) / 2MB
3. Allocate, zero, and pin a host memory page (PD page) for each SD needed from SD_index to Last_SD_index.
4. Calculate the number of PDs that need to be allocated:
   a. FPM_PD_index = (FPM_object_address / 4KB) & 0x1FF
   b. FPM_PD_limit_index = (FPM_object_limit - 1) / 4KB) & 0x1FF
   c. FPM_PD_count = FPM_PD_limit_index + 1 - FPM_PD_index
5. Initialize the PDs:
   a. Allocate/zero/pin FPM_PD_count pages (these are the FPM object backing pages).
   b. Initialize each of the PDs with the physical address of a page allocated in Step 5a and set the PD valid bit (see Table 9-12 for the format). The PDs are in the PD pages allocated in Step 3.
6. Update the SD table using the PFHMC_SDCMD, PFHMC_SDDATALOW, and PFHMC_SDDATAHIGH registers for each PD page allocated in Step 3 (if the PDs are associated with a Protocol Engine enabled VF, the GLHMC_VFSDCMD[HMC_PM_index], GLHMC_VFSDDATALOW[HMC_PM_index], and GLHMC_VFSDDATAHIGH[HMC_PM_index] registers must be used). These registers are programmed via the Update Protocol Engine SDs CQP operation for backing pages related to Protocol Engine objects.
   a. Write the most significant 32 bits of the physical address of the PD page to the PFHMC_SDDATAHIGH register.
   b. Write the last significant 32 bits of the physical address of the PD page to the PFHMC_SDDATALOW register, ensuring that the lower 12 bits are 0.
   c. Write 512 to PFHMC_SDDATALOW.*PMSDBPCOUNT*. If this was the last SD of the FPM, the value might be lower than 512, but PE QPs are in the middle of FPM space so the value must be 512. The *PMSDBPCOUNT* field is used by the E810 to calculate the end of the FPM space without having to read the valid bit for each individual PD entry.
   d. Write 0 to PFHMC_SDDATALOW.*PMSDTYPE*.
   e. Write 1 to PFHMC_SDDATALOW.*PMSDVALID*.
   f. Write PFHMC_SDCMD with *PMSDIDX* set to the proper SD index value and *PMSDWR*=1.

When this process is complete for typical configurations, the second SD is populated with the address of a single PD page, and entries 1-129 are populated with the address of the 128 FPM object backing pages that have been allocated.

9.3.8 De-Populating HMC Backing Pages

The process of de-populating and freeing HMC object backing pages is as follows:

1. Ensure that software and hardware are not going to access objects in the pages.
2. Calculate the SD range and/or PD range that provide the address mapping to the E810.
3. Update the associated PD entries.
4. Invalidate the on-die PD cache entries using the PFHMC_PDINV (or GLHMC_VFPDINV[HMC_PM_index]) register.
5. Update the SDs using the PFHMC_SDCMD, PFHMC_SDDATALOW, and PFHMC_SDDATAHIGH registers to notify the E810 that the pages are no longer valid for use. If the PDs are associated with a Protocol Engine enabled VF, the GLHMC_VFSDCMD[HMC_PM_index], GLHMC_VFSDDATALOW[HMC_PM_index], and GLHMC_VFSDDATAHIGH[HMC_PM_index] registers must be used. These registers are programmed via the Update Protocol Engine SDs CQP operation for backing pages related to Protocol Engine objects.

9.3.8.1 Removing a Backing Page

Once software has determined that a backing page is no longer needed, the software must clear the *PD_Valid* bit (see Table 9-12) in the PD entry that references the backing page. After clearing the *PD_Valid* bit in the PD in host memory, software must then write the PFHMC_PDINV register with the SD index and PD index of the newly-invalidated PD entry. This is to ensure that references to the invalid PD entry have been removed from any E810 cache. Once this write is complete, the backing page can be freed by software. The write to the PFHMC_PDINV register is not required for direct SDs since there is not a PD involved in addressing the HMC backing pages. If the backing pages are associated with a Protocol Engine enabled VF, the GLHMC_VFPDINV[HMC_PM_index] register must be used.

9.3.8.2 Removing a Page Descriptor Page

Once software has determined that an entire PD page is no longer needed, the PFHMC_SDDATALOW register must be written with PFHMC_SDDATALOW.*PMSDVALID* set to 0, and then the PFHMC_SDCMD register must be written with *PMSDIDX* set to the proper SD index value and *PMSDWR*=1. Once this sequence is complete, software is free to deallocate or re-use the PD page. If the PDs are associated with a Protocol Engine enabled VF, the GLHMC_VFSDCMD[HMC_PM_index], GLHMC_VFSDDATALOW[HMC_PM_index], and GLHMC_VFSDDATAHIGH[HMC_PM_index] registers must be used. These registers are programmed via the Update Protocol Engine SDs CQP operation for backing pages related to Protocol Engine objects.

9.3.9 Special Cases for Protocol Engine Objects

Since there are less HMC VF FPM spaces than there are VFs, there is a mapping required that is described in Section 9.3.9.1.

9.3.9.1 Virtual Function Support

The HMC supports eight HMC FPM spaces that are dedicated to PFs (index 0-7) and 32 HMC FPM spaces that are available for VFs. Since there are less HMC FPM VF spaces available than actual VFs for the E810, a mapping is required. The virtual function mapping is performed by CQP firmware using the Manage HMC PM Function Table operation found in Section 11.5.3.10.

9.3.10 HMC Error Reporting

HMC-related errors are reported through the PFHMC_ERRORINFO and PFHMC_ERRORDATA registers. The *HMC_ERR* interrupt status bit in the PFINT_OICR register can also deliver an interrupt for HMC errors if the interrupt is enabled in the PFINT_OICR_ENA register. When the HMC detects an error, it sets the PFHMC_ERRORINFO.*ERROR_DETECTED* bit along with the relevant information in the other fields of the PFHMC_ERRORINFO and PFHMC_ERRORDATA registers. No further notification of subsequent HMC errors associated with any given PF are issued until the current error is acknowledged by writing a 0 to the PFHMC_ERRORINFO.*ERROR_DETECTED* bit.

Table 9-20 describes the errors detected for each HMC object and the behavior associated with each error.

Table 9-20. HMC Errors

HMC Object	Error Type(s)	Error Behavior
Protocol Engine QP Context	• PMF Invalid • Invalid PMF Index • HMC Object Index Too Large	The index of the PE QP is reported in the PFHMC_ERRORDATA register. Packets associated with the QP are dropped. Protocol Engine Asynchronous Events are not reported.
	• HMC Private Memory Address Too Large • Segment Descriptor Invalid • Segment Descriptor Too Small • Page Descriptor Invalid	The Private Memory Address of the PE QP object is reported in the PFHMC_ERRORDATA register. Packets associated with the QP are dropped. Protocol Engine Asynchronous Events are not reported.
Protocol Engine CQ Context	• PMF Invalid • Invalid PMF Index • HMC Object Index Too Large	The index of the PE CQ is reported in the PFHMC_ERRORDATA register. Completions for all QPs associated with the CQ are dropped. Protocol Engine Asynchronous Events are not reported.
	• HMC Private Memory Address Too Large • Segment Descriptor Invalid • Segment Descriptor Too Small • Page Descriptor Invalid	The Private Memory Address of the PE CQ object is reported in the PFHMC_ERRORDATA register. Completions for all QPs associated with the CQ are dropped. Protocol Engine Asynchronous Events are not reported.
Protocol Engine TCP Timers	• PMF Invalid • Invalid PMF Index • HMC Object Index Too Large	The index of the PE timer object is reported in the PFHMC_ERRORDATA register. Timer operation associated with the function will go idle. Host software is responsible for cleaning up all Protocol Engine resources associated with the PCI function and reporting a fatal adapter error to the RDMA protocol stack.
	• HMC Private Memory Address Too Large • Segment Descriptor Invalid • Segment Descriptor Too Small • Page Descriptor Invalid	The index of the PE timer object is reported in the PFHMC_ERRORDATA register. Timer operation associated with the function will go idle. Host software is responsible for cleaning up all Protocol Engine resources associated with the PCI function and reporting a fatal adapter error to the RDMA protocol stack.

Table 9-20. HMC Errors [continued]

HMC Object	Error Type(s)	Error Behavior
Protocol Engine Hash Table Entry	• PMF Invalid • Invalid PMF Index • HMC Object Index Too Large	The index of the PE Hash Table entry is reported in the PFHMC_ERRORDATA register. Packets associated with the hash table entry are dropped. Protocol Engine Asynchronous Events are not reported.
	• HMC Private Memory Address Too Large • Segment Descriptor Invalid • Segment Descriptor Too Small • Page Descriptor Invalid	The Private Memory Address of the PE Hash Table entry object is reported in the PFHMC_ERRORDATA register. Packets associated with the hash table entry are dropped. Protocol Engine Asynchronous Events are not reported.
ARP Table Entry	• PMF Invalid • Invalid PMF Index • HMC Object Index Too Large	The index of the ARP Table entry is reported in the PFHMC_ERRORDATA register. Packets associated with the ARP table entry are dropped. Protocol Engine Asynchronous Events are not reported.
	• HMC Private Memory Address Too Large • Segment Descriptor Invalid • Segment Descriptor Too Small • Page Descriptor Invalid	The Private Memory Address of the ARP Table entry is reported in the PFHMC_ERRORDATA register. Packets associated with the ARP table entry are dropped. Protocol Engine Asynchronous Events are not reported.
Accelerated Port Bit Vector In-Use	• PMF Invalid • Invalid PMF Index • HMC Object Index Too Large	The index of the APBVT entry is reported in the PFHMC_ERRORDATA register. Packets associated with the APBVT entry are dropped. Protocol Engine Asynchronous Events are not reported.
	• HMC Private Memory Address Too Large • Segment Descriptor Invalid • Segment Descriptor Too Small • Page Descriptor Invalid	The Private Memory Address of the APBVT entry is reported in the PFHMC_ERRORDATA register. Packets associated with the APBVT entry are dropped. Protocol Engine Asynchronous Events are not reported.
Memory Region Table Entry (MRTE)	• PMF Invalid • Invalid PMF Index • HMC Object Index Too Large	The index of the MRTE is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
	• HMC Private Memory Address Too Large • Segment Descriptor Invalid • Segment Descriptor Too Small • Page Descriptor Invalid	The Private Memory Address of the MRTE object is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
Xmit FIFO	• PMF Invalid • Invalid PMF Index • HMC Object Index Too Large	The index of the Xmit FIFO entry is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
	• HMC Private Memory Address Too Large • Segment Descriptor Invalid • Segment Descriptor Too Small • Page Descriptor Invalid	The Private Memory Address of the Xmit FIFO entry is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
Xmit FIFO Free List	• PMF Invalid • Invalid PMF Index • HMC Object Index Too Large	The index of the Xmit FIFO Free List entry is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
	• HMC Private Memory Address Too Large • Segment Descriptor Invalid • Segment Descriptor Too Small • Page Descriptor Invalid	The Private Memory Address of the Xmit FIFO Free List entry is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.

Table 9-20. HMC Errors [continued]

HMC Object	Error Type(s)	Error Behavior
Inbound RDMA Read Queue (IRRQ or Q1)	• PMF Invalid • Invalid PMF Index • HMC Object Index Too Large	The index of the IRRQ entry is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
	• HMC Private Memory Address Too Large • Segment Descriptor Invalid • Segment Descriptor Too Small • Page Descriptor Invalid	The Private Memory Address of the IRRQ entry is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
Inbound RDMA Read Queue (IRRQ or Q1) Free List	• PMF Invalid • Invalid PMF Index • HMC Object Index Too Large	The index of the IRRQ Free List entry is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
	• HMC Private Memory Address Too Large • Segment Descriptor Invalid • Segment Descriptor Too Small • Page Descriptor Invalid	The Private Memory Address of the IRRQ Free List entry object is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
Multicast Group	• PMF Invalid • Invalid PMF Index • HMC Object Index Too Large	The index of the Multicast Group entry is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
	• HMC Private Memory Address Too Large • Segment Descriptor Invalid • Segment Descriptor Too Small • Page Descriptor Invalid	The Private Memory Address of the Multicast Group entry object is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
Address Handles	• PMF Invalid • Invalid PMF Index • HMC Object Index Too Large	The index of the Address Handle entry is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
	• HMC Private Memory Address Too Large • Segment Descriptor Invalid • Segment Descriptor Too Small • Page Descriptor Invalid	The Private Memory Address of the Address Handle entry object is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
Physical Buffer List Entry (PBLE)	• PMF Invalid • Invalid PMF Index • HMC Object Index Too Large	The index of the PBLE entry is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
	• HMC Private Memory Address Too Large • Segment Descriptor Invalid • Segment Descriptor Too Small • Page Descriptor Invalid	The Private Memory Address of the PBLE entry object is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
Read Response FIFO	• PMF Invalid • Invalid PMF Index • HMC Object Index Too Large	The index of the Read Response FIFO List entry is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
	• HMC Private Memory Address Too Large • Segment Descriptor Invalid • Segment Descriptor Too Small • Page Descriptor Invalid	The Private Memory Address of the Scatter Element List entry is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.

Table 9-20. HMC Errors [continued]

HMC Object	Error Type(s)	Error Behavior
Read Response FIFO Free List	• PMF Invalid • Invalid PMF Index • HMC Object Index Too Large	The index of the Read Response FIFO Free List entry is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
	• HMC Private Memory Address Too Large • Segment Descriptor Invalid • Segment Descriptor Too Small • Page Descriptor Invalid	The Private Memory Address of the STag Free List entry object is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
Header	• PMF Invalid • Invalid PMF Index • HMC Object Index Too Large	The index of the Header object is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
	• Invalid LAN Queue Index	Not Applicable to this object type.
	• HMC Private Memory Address Too Large • Segment Descriptor Invalid • Segment Descriptor Too Small • Page Descriptor Invalid	The Private Memory Address of the Header object is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
Metadata	• PMF Invalid • Invalid PMF Index • HMC Object Index Too Large	The index of the Metadata object is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
	• Invalid LAN Queue Index	Not Applicable to this object type.
	• HMC Private Memory Address Too Large • Segment Descriptor Invalid • Segment Descriptor Too Small • Page Descriptor Invalid	The Private Memory Address of the Metadata object is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
Out-of-Order Send Completion (OOISC) FIFO	• PMF Invalid • Invalid PMF Index • HMC Object Index Too Large	The index of the Out-of-Order Send Completion (OOISC) is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
	• Invalid LAN Queue Index	Not Applicable to this object type.
	• HMC Private Memory Address Too Large • Segment Descriptor Invalid • Segment Descriptor Too Small • Page Descriptor Invalid	The Private Memory Address of the Out-of-Order Send Completion (OOISC) object is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
Out-of-Order Send Completion (OOISC) FIFO Free List	• PMF Invalid • Invalid PMF Index • HMC Object Index Too Large	The index of the Out-of-Order Send Completion (OOISC) FIFO Free List entry is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.
	• Invalid LAN Queue Index	Not Applicable to this object type.
	• HMC Private Memory Address Too Large • Segment Descriptor Invalid • Segment Descriptor Too Small • Page Descriptor Invalid	The Private Memory Address of the Out-of-Order Send Completion (OOISC) FIFO Free List entry object is reported in the PFHMC_ERRORDATA register. The receive data is dropped and the QP is put into the terminate state. A Protocol Engine Asynchronous Event for the QP is reported.

9.4 Quad Hash Host Memory Cache

9.4.1 Cache Replication

The HMC for RDMA contains all the RDMA objects. This cache is partially replicated to another cache. The Quad Hash objects are duplicated for packet filtering to use. No other objects are duplicated.

When Quad Hash objects are added or deleted, the RDMA firmware is responsible for keeping the Quad Hash objects in sync for both caches. This includes populating, de-populating, and removing backing pages in both caches.

The duplicate of the Quad Hash objects cache is implemented as a separate caching block. The two caching controllers are identical design blocks. Therefore, their configuration and operation is identical as well. A separate set of registers is used for those caching blocks. Table 9-21 provides the register names of each one of those blocks.

9.4.2 Function Private Memory Space Configuration

The Quad Hash HMC configuration principles are similar to the RDMA block HMC configuration. The main difference is that Quad Hash HMC manages only one type of caching data elements. Once NVM has set the default profile or the profile was changed during the Create Control Queue Pair (Section 11.5.2.1) operation, the driver can then continue on with the second step of HMC configuration, which is to break down the FPM space into individual object regions. To do this, the driver must perform the following steps:

1. Determine the HMC function index to be configured. In the case of a PF, the HMC function number is equal to the PCI function number.
2. Read the current HMC configuration information to determine how to calculate the numbers of each HMC object that need to be requested.
    a. The GLHMC_SDPART_FPMAT[n] (or GLHMC_VFSDPART_FPMAT[n] in the case of a Protocol Engine enabled VF) register associated with the HMC FPM reports the currently available SDs for a given PCI function, which can be used to calculate the maximum FPM space.

Table 9-21 describes all the Quad HASH HMC objects and the registers used to determine the object location within Function Private Memory space, the size and limit of each object.

Table 9-21. FPM Object Registers

HMC Object	Base Register Array	Object Counter Register Array	Maximum Object Count Register	Object Element Size Register
Quad Hash Table Entry	GLHMC_PEHTEBASE_FPMAT GLHMC_VFPEHTEBASE_FPMAT	GLHMC_PEHTCNT_FPMAT GLHMC_VFPEHTCNT_FPMAT	GLHMC_PEHTMAX_FPMAT	GLHMC_PEHTEOBJSZ_FPMAT

9.4.2.1 Programming the HMC FPM Base Registers

All settings of FPM registers impact only the function associated with the registers. In other words, the driver on a given PCI function must program only the GLHMC_{object}CNT and GLHMC_{object}BASE registers for HMC PMs that are owned by that same PCI function, or HMC PMs that are associated with Protocol Engine enabled VFs that the PF owns. The FPM base of the first HMC object for each PCI function is always 0. The FPM base of subsequent HMC objects increment from previous HMC object base, the number of elements for the previous HMC object, and the size of the previous HMC object element. Additional rounding is necessary to get to the next FPM address that is properly aligned for the HMC object under consideration.

The alignment required for Hash table entry is 512B.

9.4.3 Populating HMC Backing Pages

Once the Quad Hash HMC resource profile has been picked (field) and the Function Private memory space has been programmed (Section 9.4.2), the driver must populate the HMC backing pages for the PCI function that it is initializing. The first step in this phase of initialization is to allocate 4KB pages for the PDs. Each 4KB PD page holds 512 PDs and occupies a single SD entry. Once a 4KB page has been allocated, initialized to zero, and pinned by software, the PFHMC_SDCMD_FPMAT, PFHMC_SDDATALOW_FPMAT, and PFHMC_SDDATAHIGH_FPMAT registers are used to populate the SD table for the PCI Function. Protocol Engine enabled VFs use the GLHMC_VFSDCMD_FPMAT[n], GLHMC_VFSDDATALOW_FPMAT[n], and GLHMC_SDDATAHIGH_FPMAT[n] registers, where n is the HMC Virtual Function index. Protocol Engine related SDs are programmed using the Update Protocol Engine SDs CQP operation.

Refer to the following sections:

- Section 9.3.7, “Populating HMC Backing Pages”
- Section 9.3.8, “De-Populating HMC Backing Pages”
- Section 9.3.8.1, “Removing a Backing Page”
- Section 9.3.8.2, “Removing a Page Descriptor Page”
- Section 9.3.10, “HMC Error Reporting”

9.4.4 Register Naming in Quad Hash HMC Relative to PE HMC

Table 9-22 provides control registers naming for PE HMC and Quad Hash HMC. Red text signifies the difference.

Table 9-22. Control Registers Naming

PE HMC Control Register	Quad Hash HMC Control Register
GLHMC_FWPDINV	GLHMC_FWPDINV_FPMAT
GLHMC_FWSDCMD	GLHMC_FWSDCMD_FPMAT
GLHMC_FWSDDATAHIGH	GLHMC_FWSDDATAHIGH_FPMAT
GLHMC_FWSDDATALOW	GLHMC_FWSDDATALOW_FPMAT
GLHMC_PEHTCNT_[]	GLHMC_PEHTCNT_FPMAT_[]
GLHMC_PEHTEBASE_[]	GLHMC_PEHTEBASE_FPMAT_[]
GLHMC_PEHTEOBJSZ	GLHMC_PEHTEOBJSZ_FPMAT
GLHMC_PEHTMAX	GLHMC_PEHTMAX_FPMAT
GLHMC_PFASSIGN_PMAT_[]	GLHMC_PFASSIGN_FPMAT_[]
GLHMC_PFPESDPART_[]	GLHMC_PFPESDPART_FPMAT_[]
GLHMC_PMFTABLE_PMAT_[]	GLHMC_PMFTABLE_FPMAT_[]
GLHMC_SDPART_[]	GLHMC_SDPART_FPMAT_[]
GLHMC_VFPDINV_[]	GLHMC_VFPDINV_FPMAT_[]
GLHMC_VFPEHTCNT_[]	GLHMC_VFPEHTCNT_FPMAT_[]
GLHMC_VFPEHTEBASE_[]	GLHMC_VFPEHTEBASE_FPMAT_[]
GLHMC_VFPMFMAP_PMAT_[]	GLHMC_VFPMFMAP_FPMAT_[]
GLHMC_VFPMFTABLE_PMAT_[]	GLHMC_VFPMFTABLE_FPMAT_[]
GLHMC_VFSDCMD_[]	GLHMC_VFSDCMD_FPMAT_[]
GLHMC_VFSDDATAHIGH_[]	GLHMC_VFSDDATAHIGH_FPMAT_[]
GLHMC_VFSDDATALOW_[]	GLHMC_VFSDDATALOW_FPMAT_[]
GLHMC_VFSDPART_[]	GLHMC_VFSDPART_FPMAT_[]
PF_VT_PFALLOC_PMAT_[]	PF_VT_PFALLOC_FPMAT_[]
PFHMC_PDINV_[]	PFHMC_PDINV_FPMAT_[]
PFHMC_SDCMD_[]	PFHMC_SDCMD_FPMAT_[]
PFHMC_SDDATAHIGH_[]	PFHMC_SDDATAHIGH_FPMAT_[]
PFHMC_SDDATALOW_[]	PFHMC_SDDATALOW_FPMAT_[]

9.5 Control Queues

9.5.1 Preface

The Control Queue is designed with the following goals:

- Abstract firmware interface so that firmware can be changed, whether for added functionality or bug fixes, without changing the driver. Further extension of the firmware can allow changing parts of the hardware without modifying the driver.
- Remove MMIO access from all non-essential driver paths.
- Incorporate the VF-to-PF and function-to-function mailboxes into a single, extensible interface. Shared resources should be accessed through the Control Queue.
- It is possible to write one driver that works both on a primary function and on a virtual function.
- A low resource driver, such as a pre-boot driver or an out-of-the-box driver, can use the Control Queue for limited transmit and receive.

The E810 provides three types of Control Queues:

- **Firmware Admin Queues** — For communication between the software PF driver and the the E810's firmware engine.
- **Mailbox Queues** — For communications between software drivers, either two different drivers running on the same function (PF/VF), or a PF driver communicating with its corresponding VF driver. In this case, the E810's mailbox transfers the data between the respective host memory buffers.
- **Sideband Queues** — For communications between software driver and its hardware. In this case, the E810's mailbox transfers the data between the host memory and the connected hardware device.

The E810 provides the following sets of Control Queues:

- One firmware Admin Queue per PF, used for software (E810 driver)-to-firmware communication, exposed in the PF memory BAR.
- One mailbox queue at the single PF, used for the PF-to-VF. Each mailbox queue can be used by different driver on a separate 4K memory space.
- One sideband queue per PF for the E810 driver. Sideband queues are used for the neighbor IP communication and are exposed in the PF memory BAR. Each sideband queue that can be used by a different driver is allocated with a separate 4K memory space.
- One mailbox queue per VF, used for the PF-to-VF communications and exposed in the VF memory BAR. Each mailbox queue can be used by a different driver on a separate 4K memory space.

Assumptions:

- Currently, firmware does not maintain context for any driver operation, except for the state of the queue itself.
- Firmware deals with one command at a time and does not start working on a new command before finishing its current task. This might change in a future version of the firmware, so the queue mechanism must allow for it today to avoid the need for driver modification if this happens. Firmware does, however, pipeline descriptor and data fetches to optimize execution latency.
- Mailbox queues and sideband queues are handled by hardware mailbox.

Figure 9-10 describes the control queues organization. Each queue type is represented by a colored box, with the number of instances on its right.

Rows represent the different queue types. The first row are firmware Admin Queues, the second are mailbox queues, and the third are sideband queues.

Columns represent the different functions and drivers, where each driver has its own color. We have four drivers running at the PF space and four drivers running at the VF space.

Colored lines represent messages that might pass between the mailbox queues.

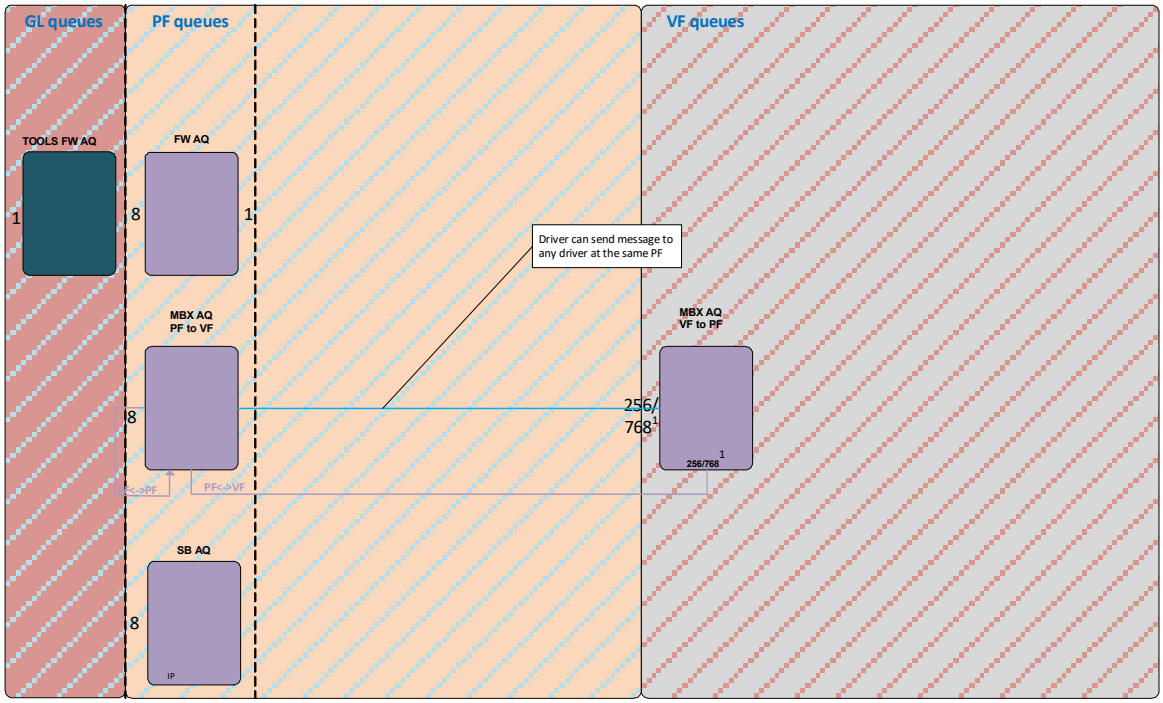


Figure 9-10. Control Queues Organization[1]

1. 768 VF queues are supported in PASID mode (GL_MBX_PASID.*PASID_MODE*).

9.5.2 Queue Structure

The Control Queue is comprised of a pair of Control Transmit Queue and Control Receive Queue. Driver commands are posted on the Control Transmit Queue (ATQ). Mailbox/Firmware completes driver commands by writing back onto the command descriptor. Events that are not an immediate result of a command are written to the Control Receive Queue (ARQ). The driver posts empty buffers to the ARQ, and the mailbox/firmware fills them with events.

Both ATQ and ARQ support direct and indirect commands, where they function as firmware or mailbox queues. Sideband ATQ queues only support indirect, as their purpose is to take the message within the buffer and send it over SB-IOSF. Sideband ARQ queues support only direct commands.

The Control Queue direct command is one that fits entirely in the queue descriptor, while an extended or indirect command is one that uses an additional buffer, which is specified in the descriptor. When a command needs an additional external buffer, it marks the *BUF* flag. If the buffer contains data that the mailbox/firmware needs to read, the *RD* flag is used. A buffer bigger than 512 bytes (AQ_LARGE_BUF) must have the *LB* flag set.

The maximum buffer size supported in this version of the queues is 4096 bytes for firmware and mailbox queues, and 512 bytes for sideband queues.

Both queues use the same 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 9-23. Control Queue 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																FE	IE	SI	BUF	VFC	RD	LB	Reserved					VFE	ERR	CMP	DD
Return Value/VFID/PFID/DRID																Datalen															
Cookie High																															
Cookie Low																															
Param0																															
Param1																															
Data Address High																															
Data Address Low																															

Table 9-24. Control Descriptor Field Descriptions

Name	Byte.Bit	Description
Flags.DD	0.0	Set by mailbox/firmware to mark entry done.
Flags.CMP	0.1	Set by mailbox/firmware to mark entry as completion.
Flags.ERR	0.2	Set by mailbox/firmware to mark entry as an error indication.
Flags.VFE	0.3	Set by mailbox/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 mailbox/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[1]
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. See Table 9-35.
Datalen	4-5	Indirect data length in bytes (can be used for other purposes if *Flags.BUF* is unset).

Table 9-24. Control Descriptor Field Descriptions [continued]

Name	Byte.Bit	Description
Return Value/VFID/ PFID/DRID	6-7	Return Value / VF ID / PF ID / DR ID for command or event. It is used by mailbox/firmware as a return value and reflects the error code as described in Section 9.5.9 and Table 9-34. When the command is used for mailbox or sideband queue, this parameter reflects the destination PF/VF/ driver ID on outgoing messages or the source PF/VF/driver ID on incoming messages.
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.
Data Address High	24-27	Indirect data pointer (can be used for other purposes if *Flags.BUF* is unset).
Data Address Low	28-31	Indirect data pointer (can be used for other purposes if *Flags.BUF* is unset).

1. *Flags.SI* is not applicable for sideband ARQ.

See Section 9.5.5 for details on the different command types.

9.5.2.1 Control Queue CSRs

The Control queues have 32 byte descriptors. They are serviced by the following registers (Table 9-25).

{PFX} denotes the register scope prefix:

- For VFs, it is "VF_"
- For PFs, it is "PF_" or "PF0_"
- For VSIs, (PASID_MODE), it is "VSI_"

Except for the name prefix the PF and VF registers are exactly the same.

There might be several control queues at the PF or at the VF context. The PF or the VF might have up to three different types of control queues (firmware, mailbox, and sideband) and up to four different queues from the same type that belong to different drivers. The queue type is added to the register name with {TYP} as follows:

- For firmware queue, it is "FW_"
- For mailbox queue, it is "MBX"
- For sideband queue, it is "SB_"

Table 9-25. Control Queue Registers

Name	Width (Bits)	Comments
{PFX}{TYP}ATQBAH	32	High bytes of ATQ base address.
{PFX}{TYP}ATQBAL	32	Low bytes of ATQ base address. Address must be 64-byte aligned.
{PFX}{TYP}ATQLEN	10 + 4	ATQ length in descriptors. MSB is set for queue enable. Three error bits (critical error, overflow error, and VF error) are set by mailbox/firmware to indicate error conditions. See Section 9.5.10.1, "Critical Error Indication" for more information.
{PFX}{TYP}ATQH	10	ATQ head pointer (mailbox/firmware updates).
{PFX}{TYP}ATQT	10	ATQ tail pointer (driver updates).
{PFX}{TYP}ARQBAH	32	High bytes of ARQ base address.

Table 9-25. Control Queue Registers [continued]

Name	Width (Bits)	Comments
{PFX}{TYP}ARQBAL	32	Low bytes of ARQ base address. Address must be 64-byte aligned.
{PFX}{TYP}ARQLEN	10 + 4	ARQ length in descriptors. MSB is set for queue enable. Three error bits (critical error, overflow error, and VF error) are set by mailbox/firmware to indicate error conditions. See Section 9.5.10.1, "Critical Error Indication" for more information.
{PFX}{TYP}ARQH	10	ARQ head pointer (mailbox/firmware updates).
{PFX}{TYP}ARQT	10	ARQ tail pointer (driver updates).

9.5.2.1.1 Control Queues CSRs Mapping

At the PF space, Control Queues targeted for different PF drivers are mapped in separate 4K pages. However, several Control Queues (for example the firmware Admin Queue and the mailbox, sideband queues) that are targeted for the same driver might be mapped on the same page. In PASID mode (set by GL_MBX_PASID.*PASID_MODE*), the control queues targeted for VSIs are mapped in separate 4K pages per VSI, at the PF space.

In non PASID mode, the same queues that were targeted for VSI are used for the virtual functions (VF). However up to 256 queues out of the 768 are in use.

See Section 13.1.2 for the exact mapping of the control queues CSRs.

9.5.2.2 Control Queue Interrupts

Firmware, mailbox, and sideband queues trigger an interrupt when they complete descriptors on the Transmit Queue or when they post events to the Receive Queue. There is a different interrupt mapping register to map the interrupts coming from firmware, mailbox, and sideband queues per driver type. Interrupts are mapped according to the following:

- Type of the control queue: firmware, mailbox, or sideband.
- The function type: PF or VF.
- The driver queue — Up to 4 different drivers, depending on the queue type and function type.

For additional details, see Section 9.1.2.5.

9.5.3 Initialization

When initializing the queue, the driver must do the following:

- The driver allocates and sets up appropriately-sized host memory for the queues.
- The driver must post initialized buffers to the Receive Queue before it can use the Transmit Queue (see Section 9.5.3.1 for Receive Queue element initialization).
- The driver clears the head and tail registers for each queue.

  Head registers are:

  {PFX}{TYP}{DRV}ATQH and {PFX}{TYP}{DRV}ARQH

  Tail registers are:

  {PFX}{TYP}{DRV}ATQT and {PFX}{TYP}{DRV}ARQT

- Then, the driver programs the base and length registers for each queue ({PFX}{TYP}{DRV}ATQBAL, {PFX}{TYP}{DRV}ATQBAH, {PFX}{TYP}{DRV}ATQLEN, {PFX}{TYP}{DRV}ARQBAL, {PFX}{TYP}{DRV}ARQBAH, and {PFX}{TYP}{DRV}ARQLEN), and sets *legnth.enable* to 1 to inform mailbox/firmware that the queue is now enabled.
- For a firmware Admin Queue, the driver must then issue a Get Version admin command and check the queue and firmware major version numbers before it can use the queue for anything else. If the major versions does not match what the driver expects, the driver reports the mismatch and fails to load. For details and current queue version number see Section 9.5.13.1, "Get Version (0x0001)".
- For a firmware Admin Queue, after the driver has verified the queue version, it sends a Driver Version command to the device. The device then sends an indication to the BMC that the PF driver is present. This is done using Host NC Driver Status Indication in NC-SI Get Link command or via the Host Network Controller Driver Status Change AEN.
- A PF driver must have at least one buffer posted to the receive queue for each VF that is currently running. This can never be greater than the number of logical cores in the system. This must be done before enabling VFs.
- A PF can disable a mailbox queue that belongs to a VF that is associated with the PF. This is done with clearing the *QUEUE_EN* bit of the relevant PF VF control register. The *QUEUE_EN* bit is allocated for each mailbox queue and there is a mailbox queue allocated for different drivers (using register name: PF_VF_MBX_CTRL, so different PF drivers can enable or disable messages from the corresponding VF driver.

9.5.3.1 Receive Queue Element Initialization by Driver

The driver must clear any unused fields (including unused flags), and set data pointers and data length to a mapped DMA pointer.

The driver can set the *SI* and the *EI* flags in the receive queue element. The driver must not set the *FE* flag on receive queue elements.

Table 9-26. Receive Queue Element - Initial Values

Name	Byte.Bit	Value	Description
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 can set	Set by driver if buffer is longer than AQ_LARGE_BUF (512 bytes).
Flags.RD	1.2	0	Not applicable to receive queue.
Flags.VFC	1.3	0	Driver must clear.
Flags.BUF	1.4	1	Receive queue elements always have an additional buffer.
Flags.SI	1.5	Driver can set	Do not interrupt when this command completes.
Flags.EI	1.6	Driver can set	Interrupt on error. Supersedes *Flags.SI* in case of error.
Flags.FE	1.7	0	Driver must clear.
Opcode	2-3		Driver must clear.
Datalen	4-5	Buffer Length	Additional data length in bytes.

Table 9-26. Receive Queue Element - Initial Values [continued]

Name	Byte.Bit	Value	Description
Return Value/VFID/ PFID/DRID	6-7		Driver must clear.
Cookie High	8-11		Driver must clear.
Cookie Low	12-15		Driver must clear.
Param0	16-19		Driver must clear.
Param1	20-23		Driver must clear.
Data Address High	24-27	Buffer Address	Indirect data pointer.
Data Address Low	28-31	Buffer Address	Indirect data pointer.

The values written by the mailbox/firmware when it uses the EAQ element are discussed in the following paragraphs.

9.5.4 Driver Unload and Queue Shutdown

When shutting down the Control Queue, the driver (both for PF and VF) does the following:

- Posts a Queue Shutdown admin command (0x0003) (Section 9.5.13.3). In this command, the driver sets the "Driver Unloading" flag if it intends to unload.
- Software must not send any additional commands on the queue until the flow is completed.

Firmware/mailbox does the following:

- Waits for any pending DMA transactions from firmware/mailbox to be acknowledged by hardware.
- Closes the Rx-Queue by clearing its "enable" bit.
- Sends a completion to the driver as usual, honoring all the interrupt control bits in the descriptor.

Software then closes the Tx-Queue by clearing its "enable" bit. (this is the *ATQENABLE* bit in the {PFX}{TYP}{DRV}_ATQLEN register).

If the driver is unloading, it issues a function reset (PFR or VFR) to the device.

If the driver is unloading, firmware informs the BMC. This is done using Host NC Driver Status Indication in NC-SI Get Link command or via the Host Network Controller Driver Status Change AEN.

- Only for a PF driver posting the Queue Shutdown command on a firmware Admin Queue.

Note: Before the Control Queues are re-enabled, software should clear the head and tail registers of the transmit and receive Control Queue.

9.5.5 Command Descriptions

9.5.5.1 Direct Command

9.5.5.1.1 Direct Admin Command

The template for a command that is fully contained in the descriptor and does not need an additional data buffer.

Table 9-27. Direct Admin Command Template

Name	Byte.Bit	Value	Description
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 has no additional write buffer.
Flags.SI	1.5	Driver can set	Do not interrupt when this command completes.
Flags.EI	1.6	Driver can set	Interrupt on error. Supersedes *Flags.SI* in case of error.
Flags.FE	1.7	Driver can set	If set, command is flushed if the preceding command resulted in an error.
Opcode	2-3	Opcode	Command opcode. See Table 9-35.
Datalen	4-5	0	
Return Value/VFID/ PFID/DRID	6-7		Mailbox messages use this field for the source/destination PF/VF/driver ID.
Cookie High	8-11	Cookie	Opaque value copied by the mailbox/firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the mailbox/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.

9.5.5.1.2 Direct Command Completion

Table 9-28. Direct Command Completion Event Template

Name	Byte.Bit	Value	Description
Flags.DD	0.0	1	Mailbox/Firmware must set.
Flags.CMP	0.1	1	Mailbox/Firmware must set.
Flags.ERR	0.2	0 or 1	Mailbox/Firmware must set only if it is reporting an error.
Flags.VFE	0.4	0	Mailbox/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	Mailbox/Firmware copies value from command.
Flags.RD	1.2	echo	Mailbox/Firmware copies value from command.
Flags.VFC	1.3	echo	Mailbox/Firmware copies value from command.
Flags.BUF	1.4	echo	Mailbox/Firmware copies value from command.
Flags.SI	1.5	echo	Mailbox/Firmware copies value from command.
Flags.EI	1.6	echo	Mailbox/Firmware copies value from command.
Flags.FE	1.7	echo	Mailbox/Firmware copies value from command.
Opcode	2-3	Opcode	Command opcode. See Table 9-35.
Datalen	4-5		Can be used for an additional command parameter.
Return Value/VFID/ PFID/DRID	6-7		Mailbox/Firmware return value 0=no error (for error codes see Table 9-34).
Cookie High	8-11	echo	Opaque value copied by the mailbox/firmware from the command.
Cookie Low	12-15	echo	Opaque value copied by the mailbox/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.

9.5.5.2 Indirect Command

9.5.5.2.1 Indirect Admin Command

An indirect write command uses an additional DMA 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 queue is limited to buffers up to 4096 bytes. If the command uses the buffer to pass data, the *RD* flag must be set.

Table 9-29. Indirect Admin Command Template

Name	Byte.Bit	Value	Description
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 can set	Set by driver if buffer is longer than AQ_LARGE_BUF (512).
Flags.RD	1.2	Driver can set	Set by driver to indicate that mailbox/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 can set	Do not interrupt when this command completes.
Flags.EI	1.6	Driver can set	Interrupt on error. Supersedes *Flags.SI* in case of error.
Flags.FE	1.7	Driver can set	If set, command is flushed if the preceding command resulted in an error.
Opcode	2-3	Opcode	Command opcode. See Table 9-35.
Datalen	4-5	Buffer Length	Usable length of additional buffer in bytes.
Return Value/VFID/ PFID/DRID	6-7		Mailbox messages use this field for the source/destination PF/VF/driver ID.
Cookie High	8-11	Cookie	Opaque value copied by the mailbox/firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the mailbox/firmware into the completion of this command.
Param0	16-19		First command parameter.
Param1	20-23		Second command parameter.
Data Address High	24-27	Buffer Address	High bits of buffer address.
Data Address Low	28-31	Buffer Address	Low bits of buffer address.

9.5.5.2.2 Indirect Command Completion

When completing an indirect command, the firmware overwrites the *Datalen* with the actual length of data returned by the command.

Table 9-30. Direct Command Completion Event Template

Name	Byte.Bit	Value	Description
Flags.DD	0.0	1	Mailbox/Firmware must set.
Flags.CMP	0.1	1	Mailbox/Firmware must set.
Flags.ERR	0.2	0 or 1	Mailbox/Firmware must set only if it is reporting an error.
Flags.VFE	0.3	0	Mailbox/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	Mailbox/Firmware copies value from command.
Flags.RD	1.2	echo	Mailbox/Firmware copies value from command.
Flags.VFC	1.3	echo	Mailbox/Firmware copies value from command.
Flags.BUF	1.4	echo	Mailbox/Firmware copies value from command.
Flags.SI	1.5	echo	Mailbox/Firmware copies value from command.
Flags.EI	1.6	echo	Mailbox/Firmware copies value from command.
Flags.FE	1.7	echo	Mailbox/Firmware copies value from command.
Opcode	2-3	Opcode	Command opcode. See Table 9-35.
Datalen	4-5	echo	Mailbox/Firmware copies value from command.
Return Value/VFID/ PFID/DRID	6-7		Mailbox/Firmware return value 0=no error (for error codes see Table 9-34).
Cookie High	8-11	echo	Opaque value copied by the mailbox/firmware from the command.
Cookie Low	12-15	echo	Opaque value copied by the mailbox/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.

9.5.6 Firmware Command Fetch and Verification

When a command is posted, firmware looks it up in an internal permission table to decide if the request should be honored. Possible actions are:

- **Allow** — Firmware acts upon the command.
- **Forward** — Firmware halts the queue and forwards the command to the PF driver. A completion for this command is initiated by the PF driver when it finishes i. Only then is further processing of commands from this queue allowed. (The PF driver must re-enable the queue after it deals with the command.)
- **Error** — Firmware completes the action by returning the error specified in the table.
- **Drop** — Firmware behaves as if the command succeed but does nothing.

9.5.7 Mailbox and Sideband Command Fetch and Verification

When a command is posted at a mailbox queue or at a sideband queue, the hardware mailbox verifies permission for the command to decide if the request should be honored. Possible actions are:

- **Allow** — Mailbox acts upon the command. In other words, it forwards the command to the relevant mailbox queue or executes the SB-IOSF transaction.
- **Error** — Mailbox completes the action by returning the error specified in the table.

Table 9-31 specifies the commands allowed in a mailbox queue and the commands allowed in a sideband queue. A PF driver is allowed to execute a richer set of commands than a VF driver. In addition, commands that deal with PF-to-VF communications are only allowed on a mailbox queue, while commands that deal with SB-IOSF are allowed on the sideband queue. If any other command is posted, the mailbox returns the EPERM (operation not permitted) error code.

Table 9-31. Commands Allowed for Mailbox and Sideband Queues

Opcode	Command	Function Allowed	Queue Type	Section Reference
0x0801	Send to PF	VF	Mailbox	9.5.14.1
0x0802	Send to VF	PF	Mailbox	9.5.14.2
0x0C00	Neighbor Device Request	PF, VF	Sideband	
0x0C01	Neighbor Device Event	PF, VF	Sideband	
0x0003	Queue Shutdown	PF, VF	Mailbox	9.5.13.3

A VF driver is only allowed to post commands at the mailbox queue, or at the sideband VF queue. Those queues are processed by the hardware mailbox. Sideband queue accepts only the Neighbor Device Request and Neighbor Device Event commands listed in Table 9-31. If any other command is posted by a VF, the mailbox returns the EPERM (operation not permitted) error code.

A PF driver can post commands at the firmware Admin Queue, Mailbox Queue, or Sideband Queue. Each queue type supports specific commands. When the command does not match the queue type, an EPERM error code is returned.

A "Queue Shutdown" command posted to a sideband queue without a buffer attached will result in an error code of EINVAL. When a buffer is attached to the "Queue Shutdown" command, the hardware will shutdown the queue.

9.5.8 Non-Completion Events

Events that are not an immediate result of a command completion are posted by the mailbox/firmware onto the receive queue.

Table 9-32 lists the currently-defined events. Note that whenever possible, the same number is used for the opcode that generates the event and for the event ID.

Table 9-32. Non-Completion Events

Opcode	Command	Type	Section Reference
0x0801	Send to PF	Indirect/Direct	9.5.14.1
0x0802	Send to VF	Indirect/Direct	9.5.14.2

9.5.9 Error Handling

When mailbox/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 9-34, and the higher byte can be used by mailbox/firmware to report internal state or debug information. The driver must log this information. Mailbox/Firmware might 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”), mailbox/firmware must not set the high byte.

Table 9-33. Control Queue Return Value Fields

Name	Bytes	Comment
Code	0-7	Return value from Table 9-34.
Mailbox/Firmware Internal Code	8-15	Mailbox/Firmware internal code. Must be 0 if *Code* field is 0.

9.5.10 Error Codes

When mailbox/firmware completes a command, it must use the following error codes.

Table 9-34. Control Queue Return Values and Error Codes

Error Code	Value	Meaning
(No Error)	0	No error (success).
EPERM	1	Operation not permitted. For mailbox queue: VF not allowed to access the PF. For sideband queue: Driver not allowed to access the sideband IP, or the driver is not allowed to send the sideband opcode.
ENOENT	2	No such element.
ESRCH	3	Bad opcode. For mailbox queue: Not a mailbox queue opcode, or incorrect use of opcode (e.g. VF uses the Send-to-VF opcode). For sideband queue: Not a sideband queue opcode.
EINTR	4	Operation interrupted.
EIO	5	I/O error.
ENXIO	6	No such resource.
E2BIG	7	Argument too long. For mailbox queue: Buffer longer than 4K bytes For sideband queue: Buffer longer than 512 bytes.
EAGAIN	8	Try again.
ENOMEM	9	Out of memory.
EACCES	10	Permission denied.

Table 9-34. Control Queue Return Values and Error Codes [continued]

Error Code	Value	Meaning
EFAULT	11	Bad address.
EBUSY	12	Device or resource busy.
EEXIST	13	Attempt to create something that exists.
EINVAL	14	Invalid argument. For mailbox queue: *Flags.BUF* and *Flags.RD* are set, but *Datalen* is equal to zero. For sideband queue: The descriptor is shorter than 8 bytes.
ENOTTY	15	Not a typewriter.
ENOSPC	16	No space left or allocation failure. For mailbox queue: No valid descriptor at the destination queue or the source buffer is bigger than the destination buffer.
ENOSYS	17	Function not implemented. For mailbox queue: Destination queue is disabled or the destination function is disabled.
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. For mailbox queue: VF greater than max supported by this function.
EFBIG	22	File too big.
ESBCOMP	23	Cannot find enough space for the message in the sideband or mailbox queue. For sideband queue: This error is flagged when the completion message from the neighbor IP was longer than 512 bytes. This error can be flagged also for sideband queues when the SB-IOSF completion from the neighbor IP was unsuccessful, or when the SB-IOSF completion contains data that could not fit the buffer size in the sideband queue buffer.
RC_ENOSEC	24	Missing security manifest.
RC_EBADSIG	25	Bad RSA signature.
RC_ESVN	26	SVN number prohibits this package.
RC_EBADMAN	27	Manifest hash mismatch.
RC_EBADBUF	28	Buffer hash mismatches manifest.
EACCES_BMCU	29	BMC update in progress. Returned when NVM ownership is required during PLDM firmware update.

9.5.10.1 Critical Error Indication

On any error that prevents data placement to a queue, the ATQLEN.*ATQCRIT* (or ARQLEN.*ARQCRIT*) bit is set by mailbox/firmware and the queue is stopped (by clearing its enable bit), then an interrupt is sent by mailbox/firmware to the driver.

Software reads and reports the error, and then resets the queue.

If an overflow occurs and a message to the queue is dropped because the queue is full, mailbox/firmware sets the ARQLEN.*ARQOVFL* bit and interrupts the driver. (Mailbox/Firmware does not stop the queue since, depending on the driver mode, this can be a recoverable error.)

Note: This error can currently only happen on the receive queue, but to simplify the hardware design, the bit is present on both queues.

When a VF has an event that causes mailbox/firmware to set an error bit in its receive queue. Mailbox/Firmware sets the ARQLEN.*ARQVFE* bit in the corresponding PF's queue and interrupts it. (Mailbox/Firmware does not stop the PF queue in this case.)

Note: Events and completions that have already been posted before the error are still readable and can be handled by software.

9.5.11 Command Opcodes

Opcodes are 16 bits. The upper eight bits designate the group of the opcode, and the lower eight bits are the command in the group. Each group is described in its relevant chapter.

Table 9-35. Opcode Groups

Name	Opcode(s)	Section Reference/Remarks
Generic	0x00xx	Section 9.5.13 and Table 9-39
MAC Address	0x0100	Section 4.2.1.3
PXE Mode	0x0110	Section 10.4.3.2
Switch	0x02xx	Section 7.8.12 and Table 7-21
DCB	0x03xx	Section 8.2.5
Scheduler	0x04xx	Section 8.3.4.3.6
RESERVED	0x05xx	Reserved.
Link	0x06xx	Section 3.2.4 and Section 3.3.8 (0x06E0:0x06F0 is dedicated to link topology commands.)
NVM	0x07xx	Section 3.4.10
Mailbox	0x08xx	Section 9.5.14
Alternate Structure	0x09xx	Section 4.3.2.2
LLDP	0x0Axx	Section 9.8.5.2.2
Receive Filters	0x0Bxx	Section 7.10.11
Sideband Control Interface	0x0C00:0x0C0F	
ACL Block	0x0C10:0x0C2F	Section 7.9.3.4
Queue Handling	0x0C30:0x0C3F	Section 10.5.5.8
Profiles Handling	0x0C40:0x0C5F	Section 7.11.9 and Section 7.11.12

9.5.12 CSR-Based Firmware Admin Queue for Tools

The E810 implements a CSR-based firmware admin queue for the use of tools. The CSR-based firmware admin queue implements a single command interface to firmware, which is CSR-based. The E810 implements a memory space, including a 32-byte descriptor for Tx and for immediate response, a 32-byte descriptor for events, and a 4KB buffer mapped into the CSR space. This area is directly accessible by the user mode tools software.

The GL_HICR and GL_HICR_EN registers are used to control the usage of this mechanism.

9.5.12.1 CSR-Based Firmware Queue Hardware Implementation

The tools software writes a single admin queue command into the 32-byte descriptor space using the same format of the admin queue command. It uses the 4K buffer as the buffer needed for the indirect command. Then, it signals the firmware using a GL_HICR CSR (as defined below) that it wrote for command execution. The firmware reads the command, executes it, and uses the same memory space to send back the answer 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.

Note: The CSRs are accessible to all PFs, but the assumption is that only PF0 will access it.

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 the 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 9-36. CSR-Based Firmware Admin Queue 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

The CSR-based firmware admin queue also implements the GL_HICR and GL_HICR_EN registers.

Table 9-37. GL_HICR

Field	Bits	Init Value	Firmware (Aux) Access	Software (CSR) Access	Description
Reserved	0	0b	RW	RW	Reserved.
C	1	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.
SV	2	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.

Table 9-37. GL_HICR [continued]

Field	Bits	Init Value	Firmware (Aux) Access	Software (CSR) Access	Description
EV	3	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.
RESERVED	31:4	0x0	RSV	RSV	Reserved.

1. Changing the *C* bit from 0b to 1b in this CSR create an interrupt to firmware. The firmware implements an interrupt routine, which reads the admin queue command from the internal memory and executes it, in a similar fashion as it executes any other admin command. The results of the admin command, is written back to the same memory.

Table 9-38. GL_HICR_EN

Field	Bits	Init Value	Firmware (Aux) Access	Software (CSR) Access	Description
EN	0	0b	RW	RO	**Enable** When set, indicates that the buffer is accessible for the device. ***Note:*** This bit is common to all functions.
RESERVED	31:1	0x0	RSV	RSV	Reserved.

9.5.12.2 Firmware and Software Interaction Using the CSR-Based Admin Queue

The following interaction is expected:

1. Firmware must register to an interrupt, which is triggered when software writes to the GL_HICR register. The mechanism is enabled by setting the GL_HICR.*EN* bit (GL_HICR.*EN* = 1).
2. Software fills the descriptor + buffer, sets the GL_HICR.*C* bit, and clears the GL_HICR.*SV* and GL_HICR.*EV* bits.
3. Once the GL_HICR.*C* bit is set, the firmware on the main CPU reads the data (descriptor + buffer), parses the packet, and checks 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. Firmware then writes a response in the 1st descriptor and optionally to the buffer, and sets the GL_HICR.*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 SW 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 32-byte 2nd descriptor, clears the GL_HICR.*C* bit, and sets the GL_HICR.*EV* bit.

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 GL_HICR.*C* bit is cleared.

Note: When the GL_HICR.*C* bit is set, the entire memory (buffer and descriptors) is RO to software.

9.5.13 Generic Firmware Admin Commands

Table 9-39. Generic Commands

Name	Opcode	Type	Section Reference
Get Version	0x0001	Direct	9.5.13.1
Driver Version	0x0002	Indirect	9.5.13.2
Queue Shutdown	0x0003	Indirect	9.5.13.3
Set PF Context	0x0004	Direct	9.5.13.4
Get Expanded AQ Error Reason	0x0005	Direct	9.5.13.5
Request Resource Ownership	0x0008	Direct	9.5.13.6
Release Resource Ownership	0x0009	Direct	9.5.13.7
Discover Function Capabilities	0x000A	Indirect	9.5.13.8
Discover Device Capabilities	0x000B	Indirect	
VM/VF Reset	0x0C31	Indirect	9.5.13.9

9.5.13.1 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 can 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 mismatches. Minor versions are for tracking changes that do not need driver modifications.

Table 9-40. Get Version Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0001	Command opcode.
Datalen	4-5	0	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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the 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.

Table 9-40. Get Version Command [continued]

Name	Byte.Bit	Value	Remarks
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).

9.5.13.2 Driver Version (0x0002)

This command is used by the driver to report the driver version. The firmware should use the driver version to report it to the BMC. See Section 9.5.3.

Table 9-41. Driver Version Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the 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 or response buffer.
Data Address Low	28-31		

Table 9-42. Driver Version Buffer

Name	Length	Description
Driver Version	Datalen	Driver string (not null terminated) as reported by driver.

9.5.13.3 Queue Shutdown (0x0003)

This is the final command posted to the queue, closing the queue as described in Section 9.5.4. When this command completes, the driver is allowed to free any host resources associated with the Control Queue.

If the driver is going to unload, it must set the *Driver Unloading* flag to inform firmware.

Once this command is posted, the driver is not allowed to issue any more commands on the queue before a reset is done.

Note: Interrupt generation and the interrupt control flags in this command are handled as usual by firmware. This means that if an interrupt was not inhibited by setting the *SI* flag, it happens. If the driver is in polling mode and can not handle an interrupt, it needs to either inhibit the interrupt or have interrupts disabled through the interrupt control registers.

Table 9-43. Queue Shutdown Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0003	Command opcode.
Datalen	4-5	0	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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Driver Unloading	16.0		1b if the driver intends to unload. 0b otherwise.
Reserved	16.1-31	0x0	Reserved.

9.5.13.4 Set PF Context (0x0004)

This admin command is used to set explicit PF ID number. It is needed for some admin commands that require control on specific PF contexts regardless of the PF from which the admin command is initiated.

Table 9-44. Set PF Context Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0004	Command opcode.
Datalen	4-5	0	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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
PF ID	16		Physical function ID.
Reserved	17-31	0x0	Reserved.

9.5.13.5 Get Expanded AQ Error Reason (0x0005)

This command is used by the software to retrieve an expanded error reason for a previous AQ command failure. This command always returns the expanded error reason for the last failed firmware Admin Queue command. This value only changes if another command fails. An additional successful command after the failed one does not change the returned value. The Get Expanded AQ Error Reason command itself does not affect the value, regardless of its success or failure. The software driver must query the firmware for the expanded error code if an AQ command returns an error and logs the *Error Reason* and *Error Identifier* values.

Table 9-45. Get Expanded AQ Error Reason Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0005	Command opcode.
Datalen	4-5	0	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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Error Reason	16-19		Error reason code. 0xFFFFFFFF indicates no additional reason specified beyond the original AQ return value.
Error Identifier	20-23		Error identifier. A zero is returned if no *Error Reason* is specified.
Reserved	24-31	0x0	Reserved.

9.5.13.6 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 9-46). On success, the command returns a returns value of 0. The completion specifies in the *Timeout* field the maximum time in milliseconds that the driver can hold the resource. 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 fails 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 9-46. Shared Resources

Resource	ID	Supported Modes	Default Timeout
NVM	0x0001	1=read, 2=write	3000 ms for read, 180000 ms for write
SDP	0x0002	1=read, 2=write	0 (no timeout)
Change Lock[1]	0x0003	1=read, 2=write	1000 ms
Global Config Lock[1]	0x0004	1=read, 2=write	3000 ms

1. This resource is used by the Download Package admin command.

Table 9-47. Request Resource Ownership Command)

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0008	Command opcode.
Datalen	4-5	0	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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Resource ID	16-17	ID	See Table 9-46 for list of IDs.
Access Type	18-19	Type	See Table 9-46 for list.
Timeout	20-23	Timeout	Timeout in milliseconds. 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 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 (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		Reserved.

Notes:

- Global Config Lock (0x0004) always returns success. Command status is reflected by the *Status* field.
- If Global Config Lock was not released and a timeout occurs, firmware issues a CORER to load working configurations from NVM.
- A software driver that successfully acquired Global Config Lock, can use only the following admin commands: Download Package, Get Version, Get Package Info List, and Release Resource Ownership with Resource ID = 0x0004. All the other commands should not be used until Global Config Lock release.
- The rest of the drivers can 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 CORER/GLOBR 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.

9.5.13.7 Release Resource Ownership (0x0009)

This command is used to return ownership of a shared resource to the firmware. The driver specifies the ID of the resource it is releasing in the *Resource ID* field.

Table 9-48. Release Resource Ownership Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0009	Command opcode.
Datalen	4-5	0	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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Resource ID	16-17	ID	See Table 9-46 for list of IDs.
Reserved	18-23		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		Reserved.

9.5.13.8 Discover Function/Device Capabilities (0x000A/ 0x000B)

This command is used to request the list of capabilities of the device or the function. The firmware fills in the capabilities structure and returns the length to the driver. If the buffer size is not big enough for the whole structure, the firmware returns ENOMEM (at the return value), writes the number of capabilities that should be returned by the command to the *Cap Count* field, and resets 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 9-50. The list of resources recognized by this version of the command are in Table 9-51.

Additional capabilities can be retrieved using the Get PHY Abilities command (0x0600) and the Get Resource Allocation command (0x0204), described in Section 3.2.4.1.4 and Section 7.8.12.2.3, respectively.

Note: Unsupported capabilities are not reported.

Table 9-49. Discover Function/Device Capabilities Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 for details.
Opcode	2-3	0x000A or 0x000B	Command opcode. 0x000A = Function 0x000B = 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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-19		Reserved.
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	

Table 9-50. Capability Structure

Name	Length (Bits)	Remarks
Capability	16	See Table 9-51 for list of capabilities.
Major Version	8	
Minor Version	8	
Number	32	Number of resources described by this capability.
Logical ID	32	Only meaningful for some types of resources.
Physical ID	32	
Additional Data1	64	In the E810, these field are used in the Logical-to-Physical Port Mapping (0x0073) capability structure.
Additional Data2	64	

Table 9-51. Resources Recognized by This Version of the Command

Name	Capability	Ver	Number	Logical ID	Physical ID
Switching Mode	0x0001	1.0	Returns the switching mode supported: 0 = EVB switching (including cloud) All other values are reserved.		
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 **Bits 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:** Reserved **Bit 4:** Reserved **Bit 5:** PLDM FW 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 7:0:** Each corresponds to a PFPCI_STATUS1.*FUNC_VALID* bit for PFs 0,…,		
Alternate RAM Structure	0x0006	1.0	1		
WoL and Proxy Support	0x0008	1.0	Number of ACPI supported filters = 8 if supported, 0 otherwise. In the E810 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 and Proxy VSI of the port. N/A if programming method is hardware. N/A for device.	**Bit 0:** 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 **Bit 2:** Proxy Support 0b = Disabled 1b = Enabled For Device command - OR of all the PFs bits. **All other bits** = Reserved.
SR-IOV	0x0012	1.0	Set to one if enabled in config space. For device, should be set if set in any of the functions.	SR-IOV version (1.1).	
Virtual Function	0x0013	1.0	Function: Number of allocated VFs. Device: Total number of VFs exposed to all functions.	Logical ID of first VF.	
VMDq	0x0014	1.0	Set to one.		
802.1Qbg	0x0015	1.0	One if enabled.		

Table 9-51. 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.	Reserved.	Reserved.
iSCSI	0x0022	1.0	0x1 if iSCSI is enabled. 0x0 if not enabled. For a device capability it is always set. For a function capability it is a reflection of the PFGEN_STATE.*PFSCEN* flag.		
RSS	0x0040	1.0	Table size: 2048 for PFs.	Entry width in bits. 8 for PFs.	
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 9-51. 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 info: **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 (GLTSYN_ENA). **Bit 3:** Reserved. **Bit 4:** Timer index owned by the function (0/1). (Note that 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_SYNCE (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. **Bits 31:25:** Reserved. Device: **Byte 0 (LSB)** - Timer ownership: **Bits 2:0:** PF owning timer 0. **Bit 3:** Timer 0 owned by a PF. **Bits 6:4:** PF owning timer 1. **Bit 7:** Timer 1 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 info: **Bit 24:**TimeSync enabled on any of the ports. **Bit 25:**Timer 0 is enabled. **Bit 26:**Timer 1 is enabled. **Bits 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 must hold the timer index for the port).

Table 9-51. Resources Recognized by This Version of the Command [continued]

Name	Capability	Ver	Number	Logical ID	Physical ID
MaxMTU	0x0047	1.0	Function: Max MTU of the function. Device: Max MTU of the hardware (9728).		
NVM Versions[2]	0x0048	1.0	1st word: NVM word address. 2nd word: NVM value.	1st word: NVM word address. 2nd word: NVM value.	1st word: NVM word address. 2nd word: NVM value.
NVM Pending Versions[3]	0x0048	1.3	1st word: NVM word address. 2nd word: NVM value.	1st word: NVM word address. 2nd word: NVM value.	1st word: NVM word address. 2nd word: NVM value.
OROM Version	0x004A	1.3	2 words - OROM version.		
OROM Pending Version[4]	0x004A	1.3	2 words - OROM version.		
Netlist Version	0x004C	1.3	BaseReleaseVersion.Major	BaseReleaseVersion.Type	BaseReleaseVersion.IANA + CustomerNetlistVersion
Netlist Pending Version[4]	0x004C	1.3	BaseReleaseVersion.Minor	BaseReleaseVersion.Type	BaseReleaseVersion.IANA + CustomerNetlistVersion
iWARP	0x0051	1.0	One if enabled		
LED[5]	0x0061	1.0	Always 1.	Action.	Pin number (GPIO Index).
SDP[6]	0x0062	1.0	Always 1.	Action.	Pin number (GPIO Index).
MDIO[7]	0x0063	1.0	Only if enabled.		The interface used to control this port.
Drop/No Drop Policy[8]	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	
Reserved	0x0071 - 0x0073	1.0	Reserved.	Reserved.	Reserved.
SKU	0x0074	1.0	Returns the SKU as described in register GL_UFUSE_SOC[15:0].	Reserved.	Reserved.
Port Mapping[9]	0x0075	1.3	Reserved	Function Number	Port Number

1. This capability is not available in the E810, 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 appears multiple times, reporting three NVM versions per entry. The following NVM words are returned as versions: 0x18-0x2E (inclusive) 0x34, and 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.
4. This capability appears only if pending image exists.
5. Repeat this entry for each assigned LED. In the E810, 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 the E810 returns an entry for each GPIO in the I/O widget, which is defined as LED in the link topology.
6. Repeat this entry for each assigned pin. In the E810, 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 the E810 returns an entry for each GPIO in the I/O widget, which is defined as SDP in the link topology.
7. This entry is not relevant for the E810 and is not returned.
8. The values in this capability are based on the PXE Mode *No-Drop Policy Supported* NVM bit and on variable sw_force_no_drop.
9. For Discover Device Capabilities, this entry is returned for each enabled function. For Discover Function Capabilities it is returned only for the current function.

Notes:

- Fields that are not applicable (empty in the table) are set to zero by the firmware.
- When device capabilities are requested, total numbers for the whole device should be returned.

9.5.13.9 VF/VM Reset (0x0C31)

In this command, software provides a list of Tx-Queue or RDMA QSet IDs that need to be closed. The AQ must be called after software stops feeding the closed queues (stop sending doorbells). RDMA QSet can be closed only when it is not associated with any QP.

The command gets the following parameters as input:

- Number of Disabled Queues/QSets
- List of Queue or QSet IDs (in the PF space)

This command deallocates nodes (in leaf layer and possibly in intermediate layers as well) in the Tx-Scheduler.

Inside the command structure, the disabled queues are organized in groups. Each group includes queues belonging to one parent node in the Tx-Scheduler structure. This organization in groups is added to ease the interface between the software and the firmware. It is not required to disable all the queues belonging to a parent in one call.

This command is called by software as part of VM/VF reset flow. When it is called as part of VM/VF reset flow, it is used for a single VM/VF per call. As part of this command flow, the EMP firmware drains the Tx-Pipe from any in-flight packets (packets scheduled for transmission by the Tx-Scheduler but have not yet been transmitted) of all disabled Tx-Queues or RDMA QSets. This is required to verify that no Tx completion is posted to software after the queue resources are released, or even re-used.

Normally, the pipe draining flow requires a very short delay. A long (or endless) Flow Control event that blocks the transmit of one or more TC's or even the entire port transmit, affects the pipe draining flow as well.

This command is completed when all in-flight packets belonging to the Disabled LAN Queues or RDMA QSets have left the Tx-Pipe. Alternately, the command is called with a timeout parameter. If the EMP firmware waits more than the timeout time, it responds with an EAGAIN error code. With the EAGAIN error code, firmware also provides a bitmap that marks which of the Congestion Domains is blocked by the long Flow Control.

Software must recall this command (with *Call-Again* flag set) prior to releasing and reusing of any of the Tx-Queues that are associated with the blocked Congestion Domains.

This command must be fully completed (all Tx-Queues or RMDA QSets released) prior to any other calling to it for other queues, and prior to calling that Move/Reconfigure Tx LAN Queues AQC.

The command's timeout is posted by software as part of the command's parameters.

Software can also instruct the EMP firmware to force the Tx-Pipe and Rx-Pipe to flush out and drop all packets from the blocked Congestion Domain. It is the PF's responsibility and authority to make the decision when to consider a long Flow Control as a malicious link partner behavior.

This is an Indirect Admin Queue command, with additional command attributes and completion attributes provided within the data buffer. Table 9-52 describes command format and defines command-specific fields.

Table 9-52. VM/VF Reset Command and Response Fields

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0C31	Same as LAN Transmit Queue Disable command (see Section 10.5.5.8.3)
Datalen	4-5		Length of response buffer.
Return Value	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Command Type and Flags	16		**Bits 0:1:** 00b = Reserved. 01b = VM Reset operation. 10b = VF Reset operation. 11b = Reserved. **Bit 2:** 0b = This is an initial call. 1b = This is a subsequent call. **Bit 3:** 0b = Return EAGAIN on timeout. 1b = Flush pipe on timeout. **Bits 4:7:** Reserved. Must set to zero. ***Note:*** Bits 2 and 3 are **NOT** mutually exclusive.
Number of Queue Groups	17		In the command: Number of Disabled Queue Groups (1..127) involved in the Q disable flow. In the response: Number of fully-processed groups.
VMVF_NUM	18-19.1		Used when the *Command Type* (Bits 0-1) is VM Reset (01b) or VF Reset (10b). Bits 0 and 1 of Byte 19 contain the MSB of the *VMVF_NUM*. This is the absolute VM or VF numbers, not relative to PF.
Timeout Time	19.2-19.7		Command timeout in units of 100 micro seconds. Valid values are 0-50.
Blocked CGDs	20	0	Zeroed by the driver. A bitmap of blocked CGDs. Set by EMP firmware when returns with EAGAIN.
Reserved	21-23		Reserved.
Data Address High	24-27	0x0	Address of buffer.
Data Address Low	28-31	0x0	

Table 9-53 describes format of the data buffer carrying additional command attributes.

Table 9-53. VM/VF Reset Command Buffer[1]

Category	Byte.Bit	Field	Description
Group #1	0-3	Parent's TEID	The TEID of the parent node to which leaves are involved.
	4	Number of Queues	(1...128).
	5	Reserved	Reserved.
	6-7.6	Queue #1	Tx-Queue ID in PF space of the first queue to be closed.
	7.7	QSet/LAN Queue	Marks if this ID points to a LAN Queue or to a RDMA QSet. 0b = LAN Queue. 1b = RDMA QSet.
	8-9.6	Queue #2	Tx-Queue ID in PF space of the second queue to be closed.
	9.7	QSet/LAN Queue	Marks if this ID points to a LAN Queue or to a RDMA QSet. 0b = LAN Queue. 1b = RDMA QSet.
	. . .		
		Queue #N	Tx-Queue ID in PF space of queue N to be closed.
		QSet/LAN Queue	Marks if this ID points to a LAN Queue or to a RDMA QSet. 0b = LAN Queue. 1b = RDMA QSet.
		Padding	Alignment to 4-byte units.
. . .			
Group #N	0-3	Parent's TEID	The TEID of the parent node to which leaves are involved.
	4	Number of Queues	(1...128).
	5	Reserved	Reserved.
	6-7.6	Queue #1	Tx-Queue ID in PF space of the first queue to be closed.
	7.7	QSet/LAN Queue	Marks if this ID points to a LAN Queue or to a RDMA QSet. 0b = LAN Queue. 1b = RDMA QSet.
	8-9.6	Queue #2	Tx-Queue ID in PF space of the second queue to be closed.
	9.7	QSet/LAN Queue	Marks if this ID points to a LAN Queue or to a RDMA QSet. 0b = LAN Queue. 1b = RDMA QSet.
	. . .		
		Queue #N	Tx-Queue ID in PF space of queue N to be closed.
		QSet/LAN Queue	Marks if this ID points to a LAN Queue or to a RDMA QSet. 0b = LAN Queue. 1b = RDMA QSet.

1. The command buffer includes a structure per involved Queues group.

9.5.13.9.1 Software Activities Prior to Calling VM/VF Reset

- Software must stop sending doorbells to the disabled queues before closing them.
- For Tx-Queues that are associated with the 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 clears 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 with the queue.
  4. When interrupt arrives, software can continue.
- In case of an VM reset or multiple queues disable, software does Step 1 for all disabled queues and Step 3 for all vectors associated to the queues.

9.5.13.9.2 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 can re-use all involved resources.
- Software can post a VM reset notification via the Completion Queue. This is done directly to each involved Completion Queue using CSRs GLCOMM_CQ_CTL[512].

9.5.14 Mailbox Commands

Table 9-54. Virtualization Admin Commands

Name	Opcode	Type	Section Reference
Send to PF	0x0801	Indirect/Direct	9.5.14.1
Send to VF	0x0802	Indirect/Direct	9.5.14.2

9.5.14.1 Send Message to PF (0x0801)

This command, together with the next one, implements a communication channel between PFs and their VFs. The data in the external buffer is copied into an event on the PF receive queue. The command completes once the data is copied. The contents of the messages are defined by software.

Since the value of the cookie is copied to the event, if eight bytes are enough for the needed message, the driver can specify a length of zero, and not use an external buffer. In this case, it should also not set the *BUF* flag.

Note: This version of the queues supports messages of up to 4096 bytes.

Table 9-55. Send to PF Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 and Section 9.5.5.2.1 for details.
Opcode	2-3	0x0801	Command opcode.
Length	4-5	Buffer Length	Length of message.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by mailbox.
Cookie High	8-11	Cookie	Opaque value copied by the mailbox/firmware into the completion of this command. It is also copied to the descriptor of the target mailbox queue.
Cookie Low	12-15	Cookie	Opaque value copied by the mailbox/firmware into the completion of this command. It is also copied to the descriptor of the target mailbox queue.
Reserved	16-23		
Data Address High	24-27	0x0	Address of data buffer
Data Address Low	28-31	0x0	

After posting the event to the PF admin receive queue, the mailbox completes this command by updating the flags and return value (0 for success).

Table 9-56. Message from VF Event

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 and Section 9.5.5.2.1 for details.
Opcode	2-3	0x0801	Event code.
Length	4-5	Buffer Length	Length of message.
Return Value/VFID	6-7	VFID	ID of sending VF. (In non-PASID mode, this is the relative virtual function ID, starting from 0 for each PF. In PASID mode, this is the absolute VSI number).
Cookie High	8-11	Cookie	Opaque value copied by the mailbox/firmware from the source descriptor.
Cookie Low	12-15	Cookie	Opaque value copied by the mailbox/firmware from the source descriptor.
Reserved	16-23		
Data Address High	24-27	0x0	Address of data buffer
Data Address Low	28-31	0x0	

9.5.14.2 Send Message to VF (0x0802)

This command, together with the previous one, implements a communication channel between PFs and their VFs. The data in the external buffer is copied into an event on the VF receive queue. The command completes once the data is copied. The contents of the messages is defined by software. A PF can only send messages to one of its VFs.

Since the value of the cookie is copied to the event, if eight bytes are enough for the needed message, the driver can specify a length of zero, and not use an external buffer. In this case, it should also not set the *BUF* flag.

Note: This version of the queues supports messages of up to 4096 bytes.

Table 9-57. Send to VF Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 and Section 9.5.5.2.1 for details.
Opcode	2-3	0x0802	Command opcode.
Length	4-5	Buffer Length	Length of message.
Return Value	6-7		Return value. Zeroed by driver. Written by mailbox.
Cookie High	8-11	Cookie	Opaque value copied by the mailbox/firmware into the completion of this command. It is also copied to the descriptor of the target mailbox queue.
Cookie Low	12-15	Cookie	Opaque value copied by the mailbox/firmware into the completion of this command. It is also copied to the descriptor of the target mailbox queue.
VFID	16-19	VFID	ID of VF. (In non-PASID mode, this is the relative virtual function ID, starting from 0 for each PF. In PASID mode this is the absolute VSI number).
Reserved	20-23		
Data Address High	24-27	0x0	Address of data buffer
Data Address Low	28-31	0x0	

After posting the event to the VF admin receive queue, the mailbox completes this command by updating the flags and return value (0 for success).

Table 9-58. Message from PF Event

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 and Section 9.5.5.2.1 for details.
Opcode	2-3	0x0802	Event code.
Length	4-5	Buffer Length	Length of message.
Return Value/VFID	6-7		Reserved.
Cookie High	8-11	Cookie	Opaque value copied by the mailbox/firmware from the source descriptor.
Cookie Low	12-15	Cookie	Opaque value copied by the mailbox/firmware from the source descriptor.
Reserved	16-23		
Data Address High	24-27	0x0	Address of data buffer
Data Address Low	28-31	0x0	

9.5.15 Software Assist Commands

9.5.15.1 Set/Get Shared Driver Parameters (0x0C90)

The Set/Get Shared Driver Parameters admin command is used by the software driver to set (write) firmware opaque parameters and get (read) them back when needed. The parameters are shared between PFs, (that is, all PFs see the same parameters).

Table 9-59. Set Shared Driver Parameters Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0C90	Command opcode.
Datalen	4-5	0	Direct command. No external buffer should be attached.
Return Value	6-7		Return value. Zeroed by driver. Written by Firmware. 0 = Success 18 = ERANGE — When the Parameter index is out of range.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command
Set/Get Opcode	16		**Bit 16.0:** Set / Get opcode 0b = Set 1b = Get **Bits 16.1-16.7:** Reserved
Parameter Index	17		Firmware maintains up to 16 32-bit wide parameters for the driver. Software should range this index between 0 and 15.
Reserved.	18-19		Reserved.
Parameter Value	20-23		32-bit value for the parameter. When the command is *Set*, software should give the value here. When the command is *Get*, the value is set by firmware at the returned descriptor.
Data Address High	24-27	0x0	Address of data buffer. Not used in this command.
Data Address Low	28-31	0x0	

Notes:

1. 1.At CORER and above, firmware initialize all parameters to 0.
2. The firmware does not maintain any lock/synchronization between PFs. The parameter holds the value set by the last *Set* command that was executed. If a *Get* is executed before any *Set*, firmware returns the initialization value (0).
3. The admin command is not provided by firmware during Firmware Recovery mode.
4. It is expected that the PF that owns the parameter resets its value in its reset flow when it goes down.
5. The *Set* command can arrive from any PF. Firmware does not limit that.

9.6 Statistics

The E810 provides statistics for the following interfaces:

- Physical ports.
- Virtual switch elements: VEBs and VSIs.
- PE statistics, some of which are global and some are per VSI.
- ACL statistics.
- Flow Director statistics.
- Host Interface statistics.

A minimal set of Ethernet interface group statistics (RFC 2863) is provided by the E810 at the switch sampling points, while a fuller set of statistics is provided for physical ports.

The prefixes for the statistics counter names are listed in Table 9-60.

Table 9-60. Counter Name Prefixes

Element	Register Prefix	Instances	Remarks
Port	GLPRT	8	
VEPA or VEB	GLSW	32	
VEB per VLAN	GL_STAT_SWR	128	
VEB per UP	GLVEBUP	32x8	Two dimensional array of 8 UPs for 32 VEBs.
VSI	GLV	768	
Switch ID	GLSWID	256	
ACL	GLSTAT_ACL	2048	
FD	GLSTAT_FD	8192	

9.6.1 Counter Implementation

Most of the counters in the E810 are used by more than one entity. For example, VSI counters are both a virtual switch port (hence used by a switch monitor agent) and a driver interface. The E810 does not provide clear-on-read statistics counters, since these would need to be duplicated per client.

As a general rule, E810 statistics counters should not be reset by software, because they are shared between more than one owner. Software must maintain a delta from the first value seen. If a software device driver needs the ability to reset counters, this should be implemented by updating the delta reference value.

To allow firmware to reset counters when a switching element is allocated, or for debug, the counters are implemented as clear-on-write, meaning that writing any value clears the counter.

The E810 provides 40-bit counters for byte and packet counters, and 32-bit counters for errors and discard events. Software should maintain counters that are appropriate in size for the OS and MIB requirement.

40-bit counters are implemented as two registers, whose names end with “high” and “low”, containing the upper eight bits and lower 32 bits, respectively.

When accessed using a 64-bit operation from PCI, these accesses are guaranteed by design to be atomic. They are internally converted to two 32-bit accesses that are always consecutive, with no interleaving between reads from different drivers. A special hardware indication tells the statistics block that this is the case. The statistics block performs one read of the counter memory to satisfy both requests, thus providing atomicity.

When 32-bit accesses are used to read the counters, each of them cause a separate read from the statistics block. Therefore, care must be taken to properly handle the possibility of one half around between the reads, since atomicity is not guaranteed.

VFs have no access to statistics registers. They should query the PF through the VF-to-PF virtual channel for their statistics.

9.6.2 Statistics Sample Points

Figure 9-11 and Figure 9-12 show sample points for Tx and Rx statistics, respectively. Error counters are in red. The figures show the processing stages in which counters are updated.

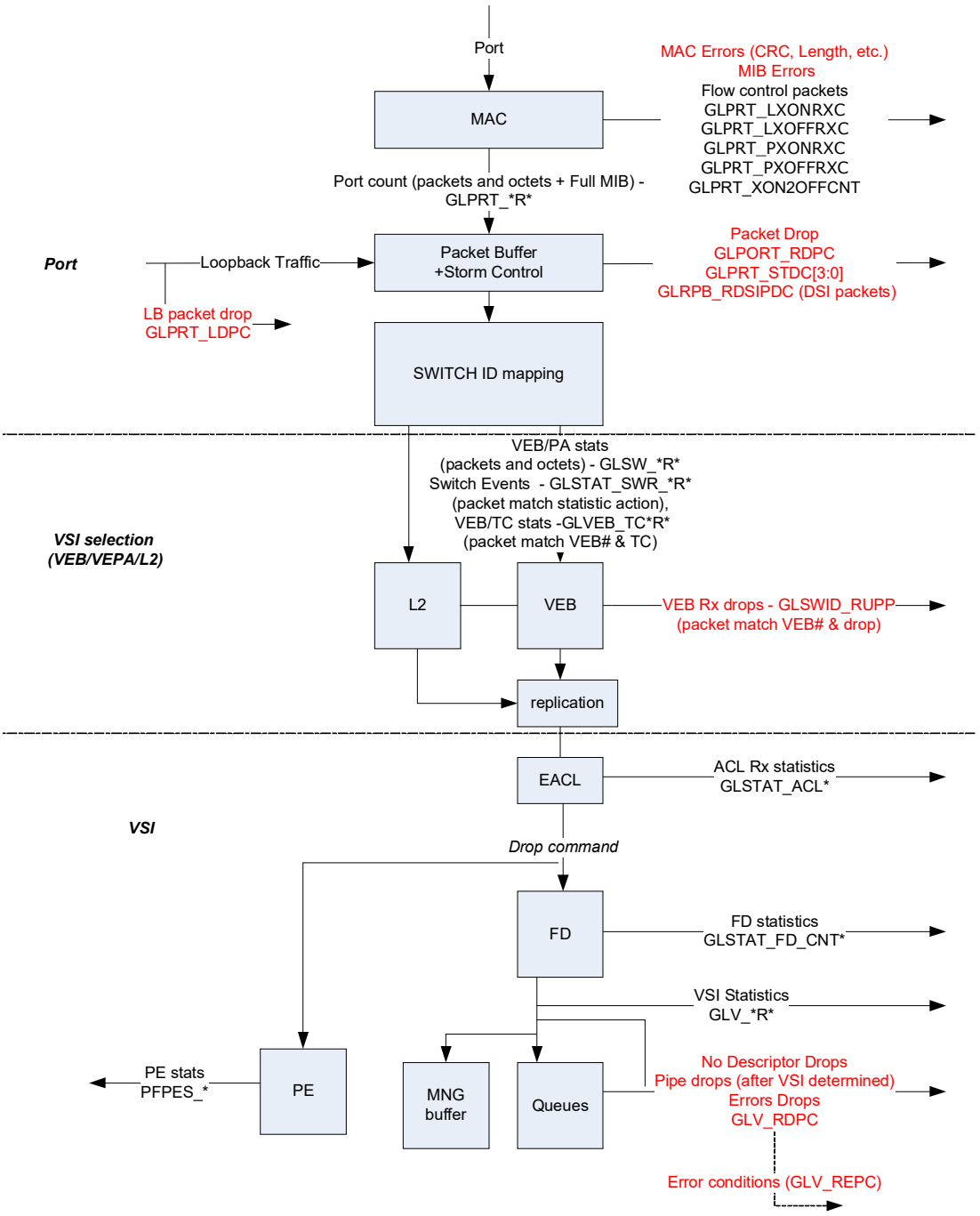


Figure 9-11. Rx Statistics Sampling Points

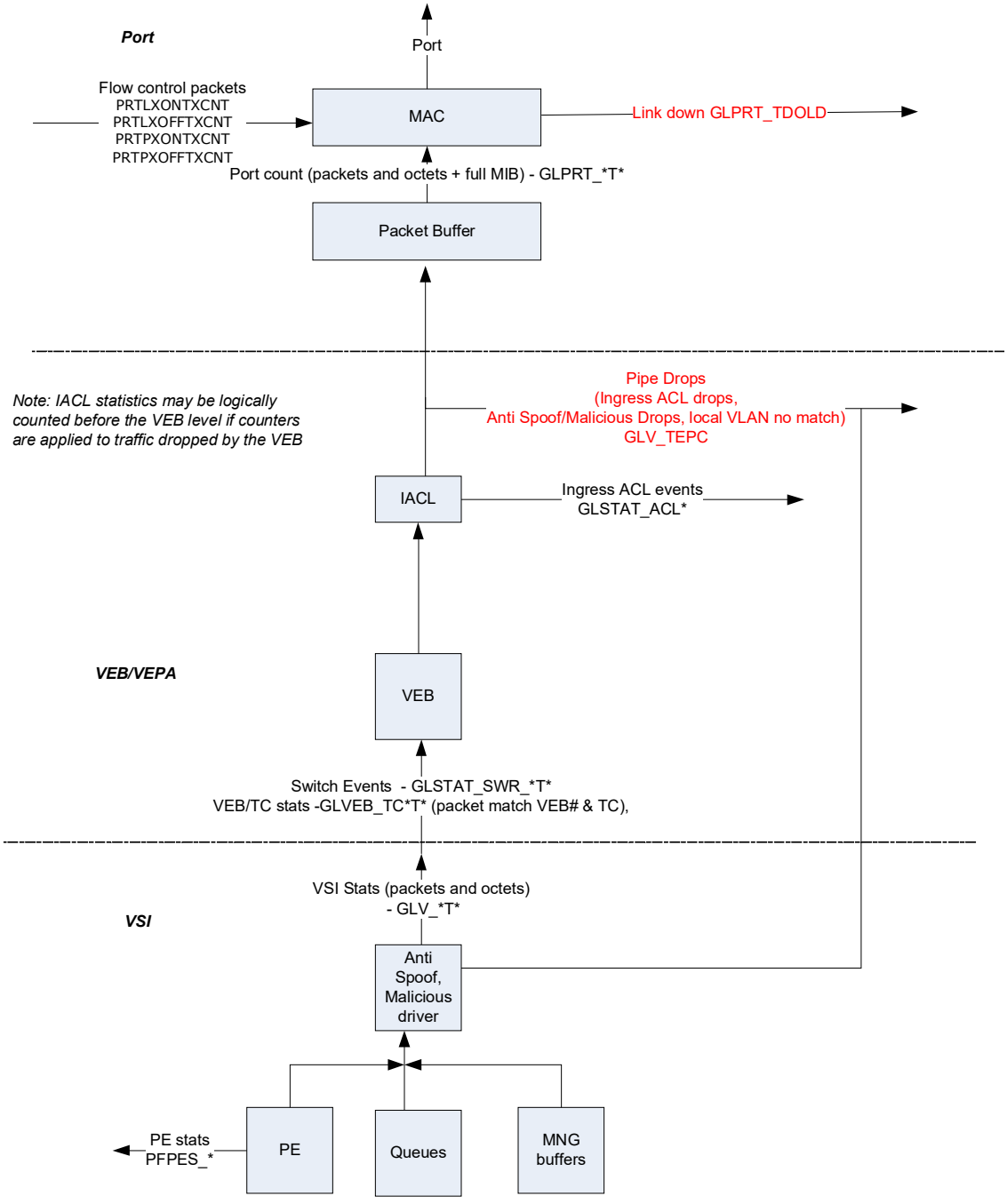


Figure 9-12. Tx Statistics Sampling Points

9.6.3 Statistics Consistency Rules

- At each switch sample point, a packet is either good or error. This means that when a packet is discarded, it is not counted in any of the non-error counters. It is also never counted by any of the next processing stages.
- All drops are counted. A packet with an error is only counted in one counter. An exception to this rule is that some debug counters are not mutually exclusive with error counters. For example, GLPRT_ERRBC and GLPRT_MSPDC are also counted as CRC errors and undersize errors, respectively.
- Ethernet octet counters count the packet as seen on the wire, from the Ethernet header up to and including the CRC (for both Rx and Tx). This means that even though an Rx packet might be stripped of tags before placement, the byte counters record the original size. Tx packets are counted after all offloads and insertions.
- Flow control packets are only counted in the flow control counters.
- Statistics specific to iWARP/RoCE count packets as they are delivered to the offload engine on Rx, and as they leave the engine on Tx. This applies to both packet lengths and packets that were merged or split. For example, packets merged by LRO are counted before the aggregation.
- Maximum lengths are adjusted to accommodate added tags. For example, the highest histogram bin MIB counter GLPRT_PTC9522 also counts any packet that would have fit in it without the added STags and VLANs.

9.6.4 Supported MIBs

The E810 supports different statistic counters as described in this chapter. The statistic counters can be used to create statistic reports as required by different standards. The E810 statistic counters allow support for the following standards:

- IEEE 802.3 clause 30 management – DTE section
- NDIS 6.0 OID_GEN_STATISTICS
- RFC 2819 – RMON Ethernet statistics group, for ports and VSIs.
- RFC 2863 – SMON Ethernet statistics group, for various switch elements.
- Linux Kernel
- net_device_stats

The following section describes the match between the internal E810 statistic counters and the counters requested by the different standards.

9.6.5 Interface Statistics at VSIs and Logical Interfaces

The E810 provides Ethernet interface group statistics per RFC 2863 on each of the Virtual Station Interfaces (VSIs), port aggregators, and virtual bridges. Table 9-61 provides an illustration of interface counters in the E810 and their sizes.

VF and PF VSIs have the same counter set. The only difference between the counter set for VSIs and the one for other switch elements is that the RUPP (unknown protocol) counter is replaced by a missed packet counter RMPC.

Note: The counter names repeat themselves with a different prefix for the element type. See Table 9-60 for the list of prefixes.

The width of the counters that are provided, is calculated to allow ample time for software to get the statistics before they wrap around. The 48-bit counters are a pair of registers, one ending with an "L" that holds with the lower 32-bits of the counter, while the higher 12 bits are in a register ending with an "H" (denoted in Table 9-61 as {H,L}). Not all counters are implemented for all prefixes. See Table 9-64 to Table 9-67 for implementation for specific prefixes.

Table 9-61. Ethernet Interface Group Statistics Counters

Register Name	Width	Description
[PFX]GORC{H,L}	40	Incoming packet octets.
[PFX]UPRC{H,L}	40	Incoming number of unicast packets.
[PFX]MPRC{H,L}	40	Incoming number of multicast packets.
[PFX]BPRC{H,L}	40	Incoming number of broadcast packets.
[PFX]RDPC	40	Incoming packet discards.
[PFX]RUPP	32	Incoming unknown protocol packets.
[PFX]GOTC{H,L}	40	Outgoing packet octets.
[PFX]UPTC{H,L}	40	Outgoing number of unicast packets.
[PFX]MPTC{H,L}	40	Outgoing number of multicast packets.
[PFX]BPTC{H,L}	40	Outgoing number of non-unicast packets.
[PFX]TDPC	32	Outgoing number of discards.
[PFX]TEPC	32	Outgoing packet errors.

Some error counters are not implemented at specific switch elements because they cannot happen at those elements in the current switch design. If software is queried about the value of these counters, it should return zero (see Table 9-62).

Table 9-62. Error Counters Not Implemented in Current Design

Counter Name	Switch Element	Description
GLPRT_XEC	Port	Port XSUM error count.
GLPRT_TEPC	Port	Port transmit error packet count (GPLRT_TDOLD counts drops due to link down).
GLV_TDPC	VSI	VSI transmit packets discarded count.
GLSW_RDPC	PA or VEB	Switch receive packets discarded count (Replaced by GLSWID_RUPP).
GLSW_TEPC	PA or VEB	Switch transmit error packet count.
GLPRT_TDPC	Port	Packets that were discarded on transmit while link was down.

9.6.5.1 MAC or Physical Uplink Interface Statistics

The MAC or Physical Uplink Interface statistics counters are accessible to the device management/ control entity in VMM or IOVM through the PF. The MAC or Physical Uplink Interface statistics use the counters defined in Table 9-63. The only exception is the Unknown Protocol Packet counter, which has no meaning in the case of the physical port and is therefor absent.

In addition, the E810 provides per MAC port some of the statistics out of the IEEE 802.3 clause 30 management counters, as well as part of the RMON Ethernet statistics group as defined by IETF RFC 2819. See Table 9-63.

Table 9-63. Additional per-Port Counters

Register Name	Width	Description
GLPRT_PRC64{H,L}	40	Packets Received [64 Bytes] Count
GLPRT_PRC127{H,L}	40	Packets Received [65–127 Bytes] Count
GLPRT_PRC255{H,L}	40	Packets Received [128–255 Bytes] Count
GLPRT_PRC511{H,L}	40	Packets Received [256–511 Bytes] Count
GLPRT_PRC1023{H,L}	40	Packets Received [512–1023 Bytes] Count
GLPRT_PRC1522{H,L}	40	Packets Received [1024 to 1522] Count
GLPRT_PRC9522{H,L}	40	Packets Received [1523 to Max Bytes] Count
GLPRT_PTC64{H,L}	40	Packets Transmitted [64 Bytes] Count
GLPRT_PTC127{H,L}	40	Packets Transmitted [65–127 Bytes] Count
GLPRT_PTC255{H,L}	40	Packets Transmitted [128–255 Bytes] Count
GLPRT_PTC511{H,L}	40	Packets Transmitted [256–511 Bytes] Count
GLPRT_PTC1023{H,L}	40	Packets Transmitted [512–1023 Bytes] Count
GLPRT_PTC1522{H,L}	40	Packets Transmitted [1024 to 1522] Count
GLPRT_PTC9522{H,L}	40	Packets Transmitted [1523 to Max Bytes] Count
GLPRT_LXONRXC{H,}	40	Link XON Received Count
GLPRT_LXOFFRXC{H,}	40	Link XOFF Received Count
GLPRT_LXONTXC{H,}	40	Link XON Transmitted Count
GLPRT_LXOFFTXC{H,}	40	Link XOFF Transmitted Count
GLPRT_PXONRXC{H,}[n]	40	Priority XON Received Count
GLPRT_PXOFFRXC{H,}[n]	40	Priority XOFF Received Count
GLPRT_PXONTXC{H,}[n]	40	Priority XON Transmitted Count
GLPRT_PXOFFTXC{H,}[n]	40	Priority XOFF Transmitted Count
GLPRT_CRCERRS{H}	40	CRC Error Count
GLPRT_ILLERRC{H,}	40	Illegal Byte Error Count
GLPRT_MLFC{H,}	40	MAC Local Fault Count
GLPRT_MRFC{H,}	40	MAC Remote Fault Count
GLPRT_RLEC{H,}	40	Receive Length Error Count
GLPRT_RUC{H,}	40	Receive Undersize Count
GLPRT_RFC{H,}	40	Receive Fragment Count
GLPRT_ROC{H,}	40	Receive Oversize Count
GLPRT_RJC{H,}	40	Receive Jabber Count
PRTRPB_RDPC	32	Received packets from the network that are dropped in the receive packet buffer. The packets are dropped due to possible lack of bandwidth on the PCIe or total bandwidth of the internal data path.
PRTRPB_LDPC	32	Same as the PRTRPB_RDPC for VM-to-VM loopback packets.
GLPRT_STDC	32	Port Storm Control Discard Count
GLPRT_TDOLD{H,}	40	Transmit Packets Dropped on Link Down

9.6.5.2 VEB Statistics

The E810 supports SMON statistics per RFC 2613 (Table 9-64) and smonPrioStats counters per priority (Table 9-65) for up to 32 VEBs. The E810 supports a set of smonVlanStats counters for up to 128 802.1Q VLANs listed in Table 9-66.

Notes:

- These VEB per VLAN counters are implemented by assigning a statistic action to a VLAN entry, and can also be used to count other types of traffic.
- When a port is using DSCP as the TC indicator, the VEB per UP counters reflects the DSCP to UP translation.
- Packets replicated by the switch are counted only once.

Table 9-64. Per-VEB Statistics Counters

Register Name	Width	Description
GLSW_GORC{H,L}	40	Incoming packet octets.
GLSW_UPRC{H,L}	40	Incoming number of unicast packets.
GLSW_MPRC{H,L}	40	Incoming number of multicast packets.
GLSW_BPRC{H,L}	40	Incoming number of broadcast packets.
GLSW_GOTC{H,L}	40	Outgoing packet octets.
GLSW_UPTC{H,L}	40	Outgoing number of unicast packets.
GLSW_MPTC{H,L}	40	Outgoing number of multicast packets.
GLSW_BPTC{H,L}	40	Outgoing number of non-unicast packets.

The following counters are available per VEB/UP:

Table 9-65. VEB per UP Counters

Register Name	Width	Description
GLVEBUP_RPC{H,L}	40	Total number of packets received per priority.
GLVEBUP_RBC{H,L}	40	Total number of octets received per priority.
GLVEBUP_TPC{H,L}	40	Total number of packets transmitted per priority.
GLVEBUP_TBC{H,L}	40	Total number of octets transmitted per priority.

Table 9-66. Event Counters

Register Name	Width	Description
GL_STAT_SWR_GORC{H,L}	40	Incoming packet octets.
GL_STAT_SWR_GOTC{H,L}	40	Outgoing packet octets.
GL_STAT_SWR_UPC{H,L}	40	Number of unicast packets
GL_STAT_SWR_MPC{H,L}	40	Number of multicast packets.
GL_STAT_SWR_BPC{H,L}	40	Number of broadcast packets.

Table 9-67. Per VSI Counters

Register Name	Width	Description
GLV_TEPC	32	Transmit error packet count.
GLV_GOTC{H,L}	40	Transmit octet count. Counts number of bytes transmitted by this VSI.
GLV_GORC{H,L}	40	Receive octet count. Counts number of bytes received by this VSI. Will count packets forwarded by switch even if dropped by later stages like ACL or Flow Director.
GLV_UPRC{H,L}	40	Receive unicast packet count. Counts number of unicast packets received by this VSI. Will count packets forwarded by switch even if dropped by later stages like ACL or Flow Director.
GLV_BPTC[H,L}	40	Transmit Broadcast packet count. Counts number of broadcast packets transmitted by this VSI.
GLV_MPTC{H,L}	40	Transmit multicast packet count. Counts number of multicast packets transmitted by this VSI.
GLV_MPRC{H,L}	40	Receive multicast packet count. Counts number of multicast packets received by this VSI. Counts packets forwarded by switch even if dropped by later stages like ACL or Flow Director.
GLV_BPRC{H,L}	40	Receive Broadcast packet count. Counts number of broadcast packets received by this VSI. Counts packets forwarded by switch even if dropped by later stages like ACL or Flow Director.
GLV_UPTC{H,L}	40	Transmit unicast packet count. Counts number of unicast packets transmitted by this VSI.
GLV_RDPC	32	Counts (per VSI) packets that were drop due to no descriptors in host queue or other drops in the pipe[1].
GLV_REPC[2]	16	*NO_DESC_CNT*[3,4] Counts drops due to no available descriptors (stuck at 0xFFFF).
	16	ERROR_CNT[3] Counts the following four error cases (stuck at 0xFFFF): 1. Packet size is larger than RXMAX of the queue. 2. Receive descriptor Unsupported Request on the PCI or internal Dummy completion. 3. Packets directed to disabled receive queues. 4. Packets dropped due to VM reset, VF reset or PF reset.

1. Pipe drops includes, for example, Flow Director, ACL, and Packets directed to invalid receive queues drops.
2. Packets dropped due to the following reasons, are not counted by REPC even if it matches one of its criteria: dropped by ACL, dropped by FD, dropped due to invalid queue.
3. These are also counted by GLV_RDPC.
4. Packets dropped and counted in GLV_REPC.*ERROR_CNT* are not counted in this counter even if descriptors are not available.

The E810 provides virtual bridge port or interface statistics counters for VSIs and uplinks associated with a VEB instance. See Section 9.6.5 for additional details on VSI and uplink interface statistics.

Notes:

- VEB statistics and VEB/TC statistics includes action-based mirror packets and manageability packets passing through this VEB. For example, traffic from BMC to host is counted both in Tx (from BMC) and Rx (to host) counters of the VEB. Port/VSI-based mirror packets are not counted in the VEB, VEB/TC counters.
- Transmits counters usually count only packets that are sent either to the LAN or to the loopback. In some cases, it might count packets that are dropped later when the packet is sent back to the receive. For example, a valid transmit packet dropped in receive due to VSI source pruning is counted in Tx counters.

9.6.5.3 ACL Statistics

The E810 provides 2048 ACL statistic counters to collect statistics based on ACL policy. ACL statistics are updated when a packet matches the ACL rule. If it matches more than one rule, only the first rule is counted as a match. ACL statistics can be used for collecting flow base statistics for accounting or billing purposes, and also to monitor network policy violations. The ACL counters are created by creating a statistic action in an ACL entry. See Section 7.9.1.6 for further details on ACL counter actions.

ACL counters are accessible via GLSTAT_ACL_CNT_n_L/H[bank_offset] registers. See Section 7.9.1.6.4 for details.

9.6.5.4 Flow Director Statistics

The E810 provides 8192 Flow Director statistic counters to collect statistics based on Flow Director actions. Flow Director statistics are updated when a packet matches a Flow Director filter and can count bytes and/or packets. Flow Director statistics can be used for collecting flow base statistics for accounting or billing purposes and also to monitor network policy violations. The Flow Director counters are addressed by creating a statistic action in an Flow Director filter. See Section 7.10.8 for further details on Flow Director counter actions.

Flow Director counters are accessible via GLSTAT_FD_CNT_L/H registers. See Section 7.10.8.1 for details.

9.6.5.5 Statistics Resources

Almost all statistic counters are statically-allocated. This means that for each element type there is a 1:1 ratio between the number of elements and the number of counter sets. The only two exceptions to this are VEB and VEB per VLAN statistics. There can be up to 256 VEBs (switch IDs), but only 32 of these can be allocated a statistic block. The total number of VLAN x VEB combinations in the system can exceed the number of counter sets.

For statically-allocated counters, the admin command allocating the object returns the offset in the counter array of the counters for this entity. For dynamically-allocated counters, the allocator must request statistics counters for the object. If statistics counters are not available, the allocation fails. In that case, the allocating driver can retry the operation without requesting statistics.

9.6.6 RDMA/RoCE Statistics

Protocol Engine (PE) statistics are defined in Section 11.9. Note that switch VSI statistics are counted for PE VSIs at PE ingress. That is, before any PE processing.

9.7 TimeSync (IEEE1588 and 802.1AS)

9.7.1 Overview

Measurement and control applications are increasingly using distributed system technologies such as network communication, local computing, and distributed objects. The 1588 standard enables accurate synchronization between clocks on distributed systems.

The E810 supports two 1588 master timers that can be synchronized to two separate 1588 time domains. The master timers are synchronized to additional 1588 timers on the PHY units. Figure 9-13 shows a top-level diagram of the 1588 logic.

PHY Timers:

The timers on the PHY units do the actual time sampling of the packets. The timers in the PHYs are synchronized to one of the two master timers by the *master_timer* parameter in the rx_timer_cmd and tx_timer_cmd PHY registers. PHY setting is per LAN port.

Master Timers:

The master timers synchronize all the timers in the PHYs. It is used by software to synchronize the time to the network time. It is used to sample and drive GPIO signals synchronized to the timer.

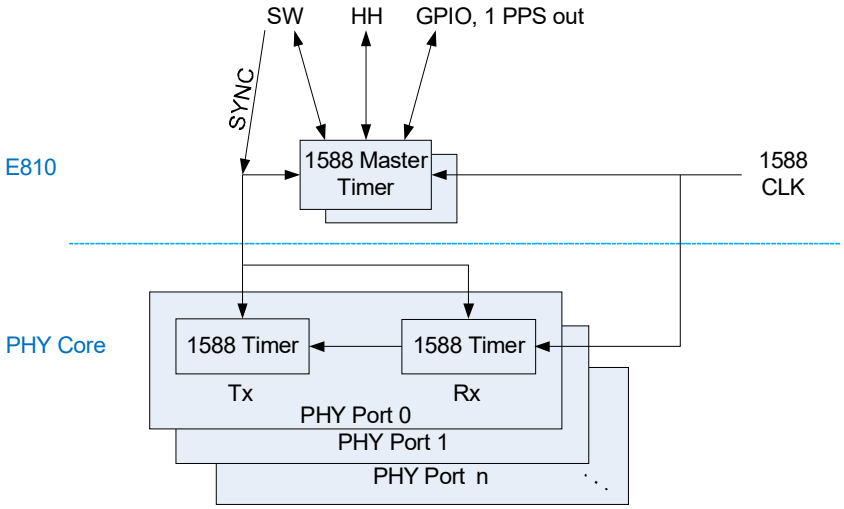


Figure 9-13. 1588 Logic - Top-Level Diagram

9.7.2 Time Synchronization - Background

9.7.2.1 Time Synchronization Flow

The operation of a 1588 logic is based on Precision Time Protocol (PTP). This protocol is composed of two stages: initialization and time synchronization. These stages are described below, emphasizing hardware and software roles.

9.7.2.2 Initialization Phase

If enabled as a potential master (by a software setting), the software periodically transmits sync packets that include the master's clock parameters. Upon receipt of a sync packet, the software on any potential master compares the received clock parameters to its own parameters. If the received parameters are better, the software transitions to a slave state and stops sending sync packets.

While in slave state, the software selects a particular master. The software continuously compares the received sync packet to its selected master. If the received sync packet belongs to a different master with better clock parameters, the software switches to the new master. Eventually only one master (with the best clock parameters) remains active, while all other nodes act as slaves listening to the single master.

Every node has a defined time-out interval. If no sync packets are received from the selected master, software on each slave switches back to the initialization phase until a new master is chosen.

Note: There is more than one option for the above flow while there are other flows that are based on static master setting. The node can be set statically to master or slave modes. While in a slave mode, the node can be tuned statically to a specific master.

9.7.2.3 Time Synchronization Phase

There are two phases to the synchronization flow: (1) the slave calibrates its clock to the master, and then (2) the master performs complete synchronization.

9.7.2.3.1 Clocks Calibration

The master sends sync packets periodically (about of 10 packets per second and up to 128 packets per second according to ITU requirements). These packets are followed by *Follow_UP* packets that indicate the transmission time. The slave captures the reception time of the sync packet, so it holds the packet transmission time at the master and its reception time at the slave.

Receiving consecutive sync packets, the slave gets the delta T of the master and can calibrate its timer to get the same delta T. During this phase, the slave adjusts the INCVAL of its 1588 timer. It can also calibrate its 1588 timer at limited accuracy in the order of the transmission delay between the master and the slave. This process is illustrated in Figure 9-14.

To minimize sampling inaccuracy, both master and slave sample the packet's transmission and reception time at a location in the hardware that has as much as possible deterministic delay from the PHY interface.

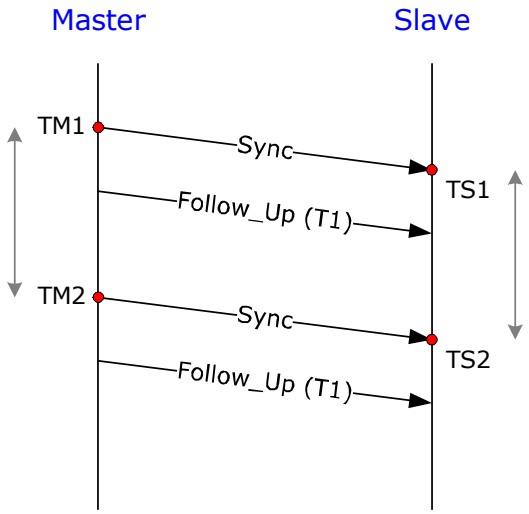


Figure 9-14. Clocks Calibration

9.7.2.3.2 Time Synchronization Phase

The complete synchronization scheme is illustrated in Figure 9-15. It relies on measured timestamp of sync packets transmission and reception by the master and the slave. The scheme is based on two assumptions:

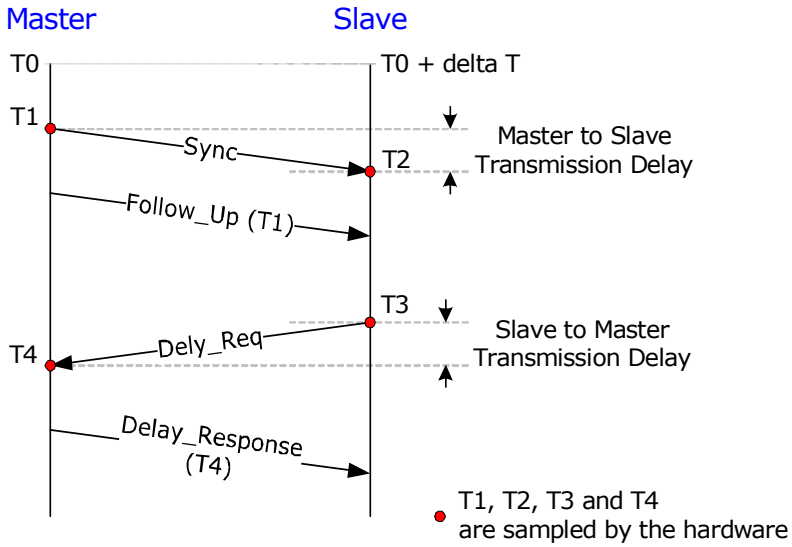
- The clocks at both nodes are almost identical (achieved in the first step).
- Transmission delays between the master to the slave and backward are symmetric. If this assumption does not hold, the software on the master as well as the slave(s) should compensate it by adjusting the sampled time of the sync packets.

The master's software sends periodic sync packets to each slave followed by a *Follow_Up* packet (as explained in Section 9.7.2.3.1). The slave samples the sync packet reception time. The slave responds back, sending a *Delay_Req* packet to the master and samples its transmission time. The master samples its reception time and reports it to the slave by a *Delay_Response* packet. At this point, the slave has all the following parameters (the notations below match the notations used in Figure 9-15):

- **T1** — *Sync* packet transmission time in the master (based on master clock)
- **T2** — *Sync* packet reception time in the slave (based on slave clock)
- **T3** — *Delay_Request* transmission time in the slave (based on slave clock)
- **T4** — *Delay_Request* reception time in the master (based on master clock)

The Slave can adjust its clock using the following equation:

Slave Adjust Time = [(T4-T3) - (T2-T1)] / 2



Calculated delta T = [(T2-T1)-(T4-T3)]/2 // assuming symmetric transmission delays

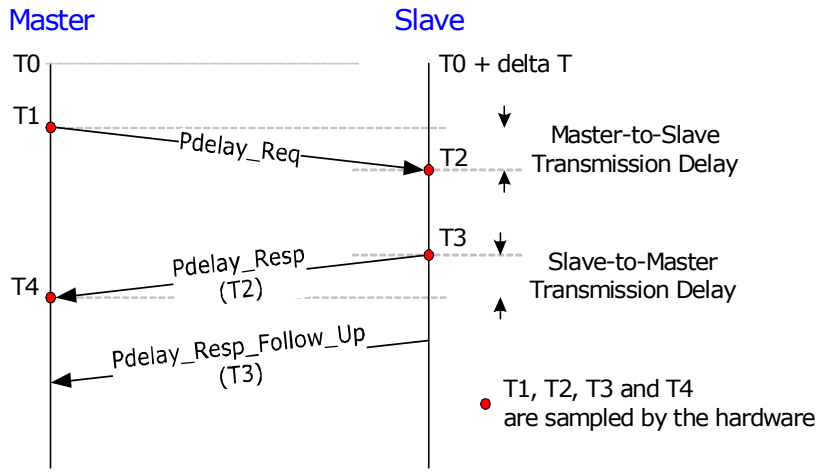
Slave Adjust Time = - delta T

Figure 9-15. Sync Flow and Offset Calculation

9.7.2.3.3 PDelay Flow for Dynamic Master Selection

The master sends a *Pdelay_Req* packet to the slave, and the slave response by sending back a *Pdelay_Resp* packet followed by a *Pdelay_Resp_Follow_Up* packet. The master uses these packets to calculate the link delay. This process is also used for fast recovery when the master is changed.

The process is usually activated when dynamic master selection is enabled. It can operate asynchronous to the "Time Synchronization" flow described in Section 9.7.2.3.2. The PDelay flow is shown in Figure 9-16.



Calculated Transmission Delay = [(T2-T1) + (T4-T3)] / 2 // assuming symmetric

Figure 9-16. PDelay Flow

9.7.3 1588 Clock and Timer Registers

9.7.3.1 1588 Master Timer Enable

The 1588 master timers are enabled by the GLTSYN_ENA register per master. When the master is not enabled, it does not increment.

9.7.3.2 Initialization

9.7.3.2.1 Firmware-Related Initialization

The EMP firmware is responsible for determining the configuration specified in this section and reflecting it to software using the Discover Function/Device Capabilities admin command (see Section 9.5.13.8). The EMP firmware must determine the mentioned configuration following each boot or following an adaptive NVM change.

9.7.3.2.1.1 Timer Ownership

Each of the two 1588 masters can be controlled by a single PF driver. The ownership is determined by an NVM-loaded configuration that can be modified by the adaptive NVM procedure (see Section 6.1.6).

9.7.3.2.1.2 Association of Timer to PHY(s)

Each PHY can be associated to one of the two master timers. The ownership is determined by an NVM-loaded configuration that can be modified by the adaptive NVM procedure (see Section 6.1.6).

9.7.3.2.1.3 Association of GPIOs to Timers

As mentioned in Section 9.7.6.1, each master timer can be associated with GPIOs. The ownership is determined by an NVM-loaded configuration.

9.7.3.2.1.4 1588 Source Clock

The 1588 timers in the E810 and the PHYs operate on the same clock of 812.5 MHz.

9.7.3.2.2 Software-Related Initialization

As mentioned in Section 9.7.3.2.1, the PF driver gets the following information using the Discover Function/Device Capabilities admin command (see Section 9.5.13.8):

- The PF driver ownership on 1588 master timers (see Section 9.7.3.2.1.1).
- The association of 1588 timers to PHY(s) (see Section 9.7.3.2.1.2).
- The association of GPIOs to 1588 master timers (see Section 9.7.3.2.1.3).
- Information about the 1588 source clock (see Section 9.7.3.2.1.4).

The PF driver that owns a 1588 master is expected to initialize and program during run time the 1588 master timer and its associated PHYs. As part of the PF initialization sequence, the driver of the PF that owns the 1588 master initializes the clock logic in the PHYs that it controls. The driver reports the PLL clock frequency to the PHYs.

The PF driver also sets the *SEL_MASTER* to one of the two 1588 masters.

9.7.3.3 1588 Timer Registers

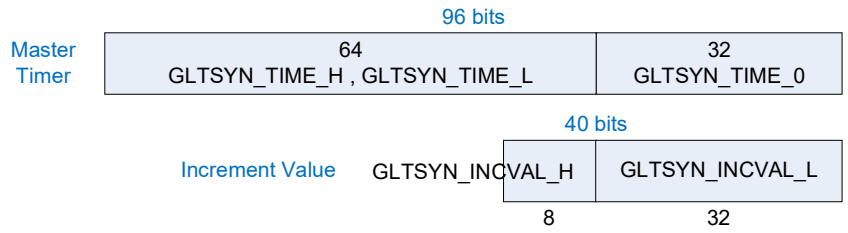


Figure 9-17. 1588 Timers

As shown in Figure 9-17, the 1588 timers are composed of two sets of registers: the timer registers and the step registers:

GLTSYN_TIME:

The E810 has two 1588 master timers, each of them is a 96-bit counter: GLTSYN_SHTIME_H, GLTSYN_SHTIME_L, and GLTSYN_SHTIME_0. Each 1588 clock, the 1588 timers are increment by a programmable "step" called INCVAL. Setting the INCVAL as specified in Table 9-68, the upper 64 bits of the timer represent the time in nanosecond units.

The PHYs have dedicated timer registers for each transmit and each receive port. The timers in the PHYs are 64-bit registers that match the lower 64-bits of the master timer in the E810.

During nominal operation, the timers in the PHYs always show the exact values as its master timer in the E810. This synchronization is achieved by initializing the master timer and the PHY's timers, including its INCVAL registers, to the same values at the same time. Setting the master timer and the PHY timers with the same values is done using the Read/Write 1588 PHY sideband admin command (the setting flows are detailed in the following sections).

The software can read the master timer value(s) by the Read 1588 PHY sideband admin command. It can also read the upper 64 bits of the timer by a direct access to the timer registers in the E810 in the following order: Read first the GLTSYN_TIME_L register and then the GLTSYN_TIME_H register. The software should re-read the GLTSYN_TIME_L register to ensure that it was not wrapped around before the read of the GLTSYN_TIME_H register. If it does wrap around, software should start the read process again.

GLTSYN_INCVAL:

As indicated above, at each 1588 clock pulse the 1588 timers are increment by a programmable "step" called INCVAL. The INCVAL is a 40-bit field. The E810 contains the INCVAL in the GLTSYN_INCVAL_L and GLTSYN_INCVAL_H registers per master timer. The PHYs also have the same 40-bit INCVAL registers per transmit and receive timers per port.

During nominal operation, the software programs the INCVAL of the master timer and the INCVAL of the PHYs together by the Read/Write 1588 PHY sideband admin command.

1588 Clock Frequencies and Its Matched INCVAL:

Table 9-68 shows the INCVAL settings by which the upper 64-bits of the timer represent the time in nanosecond units. These settings (done by the software) are just an example for the INCVAL values if the timer operates as asynchronous timer. When the 1588 timer is synchronized to another timer, the INCVAL is adjusted during time according to the difference between the local clock frequency relative to the clock frequency of the other timer.

Note: There are applications on which the timer represents the time in units of its input clock. In this case, the INCVAL in the master node in the network is set to 0x100000000 and there is no need for any adjustment as indicated in the last column in Table 9-68.

Table 9-68. Recommended INCVAL for a Given PLL Frequency

Source Clock	Source Clock Frequency (MHz)	Matched PLL Frequency (MHz)	INCVAL for Getting a Timer in Nanosecond units	Time Adjust in Sub-Nanosecond Units for Accurate Asynchronous Clock
TIME_REF	812.5 MHz	812.5 MHz	1	0x13B13B13B

Time Adjustment and GLTSYN_SHADJ:

Adjusting the timer by 'N' or by +-1: On each 1588 clock, the timer is incremented by INCVAL. When adjusting the timer by 'N' or by +-1, the timer is incremented by the INCVAL + ('N' or +-1) for one 1588 clock and then it reverts back to incremental step of INCVAL.

Note that +- 1 equals to +- 0x100000000 (one and 32 hex zero's).

Note: The 'N' in the equation above is the value of the *ADJUST* field in the GLTSYN_SHADJ registers. Negative numbers are represented in 2's complement.

9.7.4 Programming the 1588 Timers

As indicated, the E810 supports two 1588 master timers, plus the timers in the PHYs that are synchronized to one of the master timers. This section describes how the software sets the timer registers and the mechanism by which the PHY timers are synchronized to one of the master timers.

The synchronization mechanism is based on two pillars:

- All timers (master and PHY timer) operate on the same clock.
- The registers of all timers are programmed to the same values at the exact same clock.
  - All the timers that should be programmed by a specific parameter are pre-armed by the required instruction.

Then the *SYNC* signals are driven, which executes the pre-armed instruction to all selected timers on the same clock. See the timing diagram in the Figure 9-18.

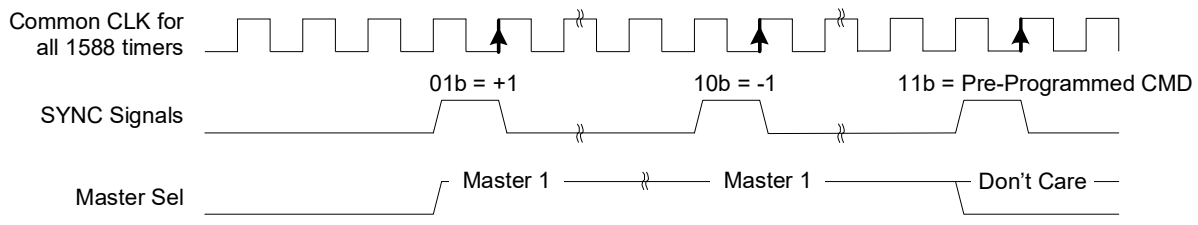


Figure 9-18. SYNC Timing Diagram

It is assumed that only one PF manages one of the 1588 masters and the timers in the PHYs that should be synchronized to that master. Programming or reading some parameters are composed of a sequence of steps. The whole sequence should be executed as an atomic action. Otherwise, possible interleaved sequences would result in an unexpected response. The atomic operation is achieved by using a semaphore scheme explained in Section 9.7.4.1.

Table 9-69 lists the timers' programming primitives that the master timer and PHY timers support:

Table 9-69. Programming the 1588 Timer Primitives

Programming Primitive Supported by the PHYs Controlled by the Sideband Messages		
No action	Init Value of the Inc Value / Timer Value	Adjust the Timer by 'N'
Sample the Timer and the Inc Value	Increment / Decrement the Timer by 1	Adjust the Timer by 'N' at a specific time

9.7.4.1 Semaphore Scheme Enabling Atomic Sequences

The E810 includes a *BUSY* flag in the PFTSYN_SEM register that can be used to gain control of a shared hardware resources. It can be used by a software driver to access the device for a sequence of actions without possible interference by other software drivers (possibly on other PFs).

1. Software should read the PFTSYN_SEM.*BUSY* flag.
2. Reading a *BUSY* flag value of zero (0b) means that the software can access the hardware.
3. Following this read access, the *BUSY* flag is auto-set by the hardware and the PF index (that initiated the read access) is captured in the PFTSYN_SEM.*PF_OWNER* field.
4. Any following read access to the register shows an active *BUSY* that tells the software it cannot access the shared hardware resource.
5. It is the responsibility of software to keep the *BUSY* active as short as possible. Once the software completes the sequence that must be atomic, it should write to clear the *BUSY* flag.
6. On a PF reset event, the EMP firmware must read the PFTSYN_SEM register and, in case of a PF reset to the PF whose index is captured in PFTSYN_SEM.*PF_OWNER* and while the *BUSY* indication is set, the EMP firmware must automatically clear the BUSY indication to allow other PFs to access the registers.

9.7.4.2 Shadow Registers for Timer Programming

Table 9-69 lists the timers' programming primitives. Supporting these primitives, each timer (masters and slaves) holds the following shadow registers.

- **Shadow Time** — These registers are used for one of the following actions:
  - — Set initial value of the timer.
  - — Sample the timer registers.
  - — Set the time for the "Adjust the Timer by 'N' at a specific time" command.

  In the E810, it is a 96-bit register per timer, called GLTSYN_SHTIME. In the PHY Core, these registers are 64 bits long (two sets of registers in the PHY Core: for the Tx and Rx data paths).
- **Shadow Adjust** — These registers are used for one of the following actions:
  - — Set initial value of the INCVAL registers.
  - — Sample the INCVAL registers.

— Adjust the timer registers.

In the E810, it is a 48-bit register per timer, called GLTSYN_SHADJ. Same register size in the PHY Core (two sets of registers in the PHY Core: for the Tx and Rx data paths).

- **CMD Register 1** — The PHY Core has two sets of command registers: rx_master_timer and tx_master_timer. These registers are programmed by the command to be executed at *SYNC* pulse. It contains the 3-bit opcode and one bit selecting the 1588 master timer. At any write access, the associated master timer should be set (not indicated explicitly in the following subsections).
- **CMD Register 2** — The E810 have two command registers used to control the 1588 timers:
  - **GLTSYN_CMD** — Contains the command to be executed and a select the master timer flag.
  - **GLTSYN_CMD_SYNC** — Writing to this register generates the *SYNC* pulse.

9.7.4.3 Initializing the 1588 Timers and the INCVAL

This sequence is useful mainly at timer initialization. It makes sense to initialize both the timer values and its INCVAL in the same step, as shown below. This is the first step that group a set of PHYs with one of the two master timers in the E810.

1. Read the PFTSYN_SEM.*BUSY* flag until it is zero (0b).
2. Program the CMD registers and the shadow registers of all the relevant PHY ports that should be tied to the selected master using the Read/Write 1588 PHY sideband Admin Queue command with the following parameters (both the Tx and Rx PHY registers):
   a. CMD register = INIT INCVAL + TIMER at SYNC
   b. Set the SHTIME_L register (32 bits) // Equivalent to GLTSYN_SHTIME_0 of the master.
   c. Set the SHTIME_H register (32 bits) // Equivalent to GLTSYN_SHTIME_L of the master.
   d. Set the SHADJ_L register (32 bits) // Same as GLTSYN_SHADJ_L of the master.
   e. Set the SHADJ_H register (32 bits) // Same as GLTSYN_SHADJ_H of the master.
3. Set the GLTSYN_SHTIME (0, L, and H) and GLTSYN_SHADJ (L and H) registers of the master timer in the E810.
4. Set the *CMD* to INIT INCVAL + TIMER and the *SEL_MASTER* as needed in the GLTSYN_CMD register in the E810. The *Sel_Master* signal is driven to the master and the PHYs.
5. Set the *SYNC* field in the GLTSYN_CMD_SYNC register to 11b.
   a. As a response, the E810 drives the sync signals to the masters and the PHYs.
   b. The programmed values are loaded to the master timer and the PHY timers: The SHTIME registers are loaded to the TIME registers and the SHADJ registers are loaded to the INCVAL registers.
   c. After the sync signals are generated, the *SYNC* field is auto-cleared in the GLTSYN_CMD_SYNC register.
6. Clear the PFTSYN_SEM.*BUSY* flag, enabling other software drivers to access the 1588 logic.

Note: In Step 2, the software can indicate the admin commands for all registers of all the relevant PHYs by a single tail bump of the sideband AQ. The E810 drives these sideband commands to the PHYs one-by-one, driving the next command in line only after the previous one is completed. This comment is relevant to all the following flows accessing the PHYs.

9.7.4.4 Adjust the Timer by 'N'

This step is useful at run time when the timer should be adjusted by a larger value than one LS bit of the GLTSYN_TIME_L or more granular value.

1. Read the PFTSYN_SEM.*BUSY* flag until it is zero (0b).
2. Program the CMD registers and the shadow registers of all the relevant PHY ports that should be tied to the selected master using the Read/Write 1588 PHY sideband Admin Queue command with the following parameters (both the Tx and Rx PHY registers):
   a. CMD register = ADJUST at SYNC
   b. Set the SHADJ_L register (32 bits).
   c. Set the SHADJ_H register (32 bits).
3. Set the GLTSYN_SHADJ registers of the master timer in the E810.
4. Set the *CMD* to ADJUST and set the *SEL_MASTER* as required in the GLTSYN_CMD register in the E810. The *Sel_Master* signal is driven to the master and the PHYs.
5. Set the *SYNC* field in the GLTSYN_CMD_SYNC register to 11b.
   a. As a response, the E810 drives the sync signals to the masters and the PHYs.
   b. The TIME registers are updated by the SHADJ value (on top of the INCVAL) in the master timer and the selected PHYs.
   c. After the sync signals are generated, the *SYNC* field is auto-cleared in the GLTSYN_CMD_SYNC register.
6. Clear the PFTSYN_SEM.*BUSY* flag, enabling other software drivers to access the 1588 logic.

Notes: See note in Section 9.7.4.3 for possible tail bump of the sideband AQ.

This command can only be used to increment the timer. When decrementing the timer is required, the software should set the SHADJ value to the 2's complement value of the requested decrement operand.

9.7.4.5 Adjust the Timer by 'N' at Time

This step is very similar to the adjust time sequence, plus an option to execute this step at a specific time (when the timers cross the value defined by the time parameter).

Note: This timer manipulation action is designed to be performed in a future time (hence the "at time" term). The hardware assumes that a new operation for the timer is not performed until this action is executed. Furthermore, if any of the CSRs holding settings mentioned in the flow below are changed before the timer manipulation executes, the new values are used.

1. Read the PFTSYN_SEM.*BUSY* flag until it is zero (0b).
2. Program the CMD registers and the shadow registers of all the relevant PHY ports that should be tied to the selected master using the Read/Write 1588 PHY sideband Admin Queue command with the following parameters (both the Tx and Rx PHY registers):
   a. CMD register = ADJUST at Time after SYNC
   b. Set the SHTIME_L register to the target time low // Equivalent to GLTSYN_SHTIME_0.
   c. Set the SHTIME_H register to the target time high // Equivalent to GLTSYN_SHTIME_L.
   d. Set the SHADJ_L register (32 bits).

e. Set the SHADJ_H register (32 bits).

3. Set the GLTSYN_SHTIME and GLTSYN_SHADJ registers of the master timer in the E810:
   a. Set the target time to the GLTSYN_SHTIME_0 and GLTSYN_SHTIME_L registers. The hardware ignores the value of the 32 MS bits of the timer and the GLTSYN_SHTIME_H register in this command.
   b. Set the adjust value to the GLTSYN_SHADJ_L and GLTSYN_SHADJ_H registers.
4. Set the *CMD* to ADJUST at time and set the *SEL_MASTER* as required in the GLTSYN_CMD register in the E810. The *Sel_Master* signal is driven to the master and the PHYs.
5. Set the *SYNC* field in the GLTSYN_CMD_SYNC register to 11b.
   a. As a response, the E810 drives the sync signals to the masters and the PHYs.
   b. The TIME registers are updated by the SHADJ value (on top of the INCVAL) in the master timer and the selected PHYs, one clock after the TIME registers cross the SHTIME registers. Note that if the TIME registers are exactly equal to the SHTIME registers at sync time, the TIME is updated by the SHADJ value only after the next time it cross the SHTIME registers.
   c. After the sync signals are generated, the *SYNC* field is auto-cleared in the GLTSYN_CMD_SYNC register.
6. Clear the PFTSYN_SEM.*BUSY* flag, enabling other software drivers to access the 1588 logic.

Notes: See note in Section 9.7.4.3 for possible tail bump of the sideband AQ.

This command can only be used to increment the timer. When decrementing the timer is required, the software should set the SHADJ value to the 2's complement value of the requested decrement operand.

Once the shadow time registers are set using this flow, wait for this flow to complete and for the timer to be updated before overriding the shadow time registers. That is, the shadow time registers should not be overridden before the timer update time occurs.

9.7.4.6 Read the Timer Values

Software on any PF can read the master timer and its INCVAL by direct read access to the GLTSYN_TIME and GLTSYN_INCVAL registers. Furthermore, the software driver that controls the timer can sample the timer and its INCVAL of both masters and the PHY timers. Checking the PHY timers against their master can be useful for debug purposes.

Software can also sample both master timers. It can be useful for applications that care about 2x 1588 time domains and the time delta between the two domains.

The following flow describes the steps to sample the time and INCVAL of the timers.

1. Read the PFTSYN_SEM.*BUSY* flag until it is zero (0b).
2. Optional: Program the CMD registers of all the relevant PHY ports that should be sampled by the Read/Write 1588 PHY sideband Admin Queue command with the following parameters:
   a. CMD = SAMPLE TIME at SYNC and SEL_TX_RX (selecting the Tx or Rx timer to be sampled)
3. Set the *CMD* to READ_TIME in the GLTSYN_CMD register. The *SEL_MASTER* field value is don't care.
4. Set the *SYNC* field in the GLTSYN_CMD_SYNC register to 11b.
   a. As a response, the E810 drives the Sync signals to the masters and the PHYs.

b. The 1588 timers and the INCVAL of the selected PHYs are sampled by their SHTIME and SHADJ registers, respectively. The master timers and INCVAL are sampled by their GLTSYN_SHTIME and GLTSYN_SHADJ registers, respectively.

5. Software can read the sampled TIME of both masters in their GLTSYN_SHTIME registers, and read the sampled INCVAL in their GLTSYN_INCVAL registers.
6. Software can read the sampled TIME and INCVAL registers of the PHYs that are tied to the selected master by the Read 1588 PHY sideband Admin Queue command.
7. Clear the PFTSYN_SEM.*BUSY* flag, enabling other software drivers to access the 1588 logic.

Note: See note in Section 9.7.4.3 for possible tail bump of the sideband AQ.

9.7.5 Timestamp Indication

Packet transmission time is sampled by the PHY logic at a deterministic as possible affinity to the link interface. The sampled time is taken at the beginning of the packet, as shown in Figure 9-19.

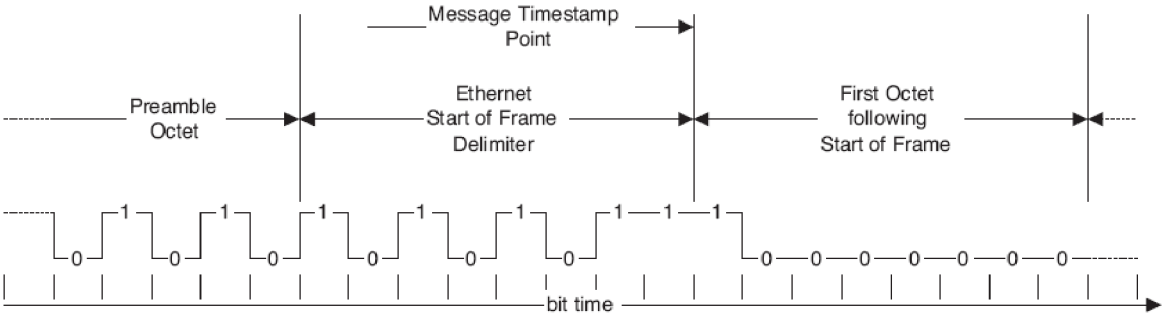


Figure 9-19. Timestamp Point

The sampled time in the PHY is a 39-bit word plus a valid indication. These 39 bits are composed of the PHY_TIME_H value plus the seven MS bits of the PHY_TIME_L (usually the sub-nanosecond units). It is illustrated in the Figure 9-20.

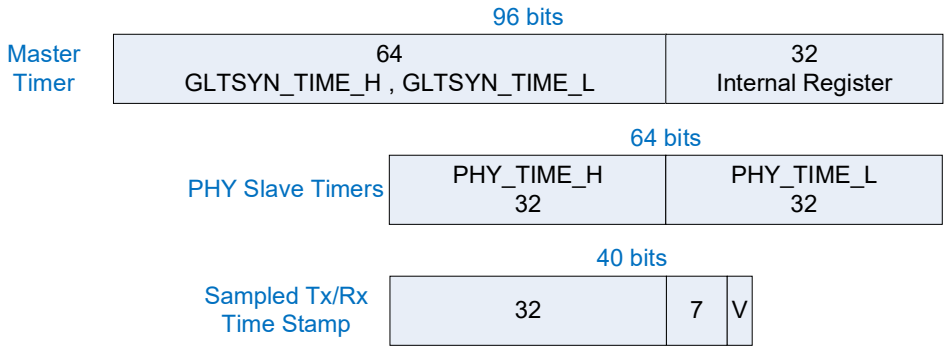


Figure 9-20. Sampled Timestamp

9.7.5.1 Transmit Timestamp

Software indicates to the hardware the packets to be sampled by setting the *TSYN* flag and the *TSYN_REG* field in the transmit context descriptor (see Section 10.5.3.2). The hardware samples the transmission time of packets with an active *TSYN* bit. Setting the *TSYN* flag in the context descriptor is meaningful only from those queues that are enabled for time sampling. A queue is enabled to use the TimeSync features using the *TSYN_ENA* flag in the transmit queue context (see Section 10.5.5.2.1).

Transmit packets are forwarded to the PHY with the *TSYN* flag and the *TSYN_REG* field as part of the packet metadata. The PHY samples the packet transmission time in a register index equals *TSYN_REG*. Once the packet is transmitted and its time is sampled, the PHY triggers a *Tx_Sample* pulse. If the specific PHY is enabled for the PF by the PFINT_TSYN_MSK register, the *1588_TX* flag in the other cause interrupt register is set, generating an interrupt to the software.

Then, the software can read the timestamp registers of all its transmitted packets that might be completed (as shown in Figure 9-21). Software reads the sampled timestamps by the Read 1588 PHY admin command (reading the low and high portions of the sampled time registers in the PHY).

Note: It is software's responsibility to read the sampled time before overriding these values by new packets. The PF driver should enable time sampling for those transmit queues on which it is guaranteed that the software is well behaved so the above rule is kept.

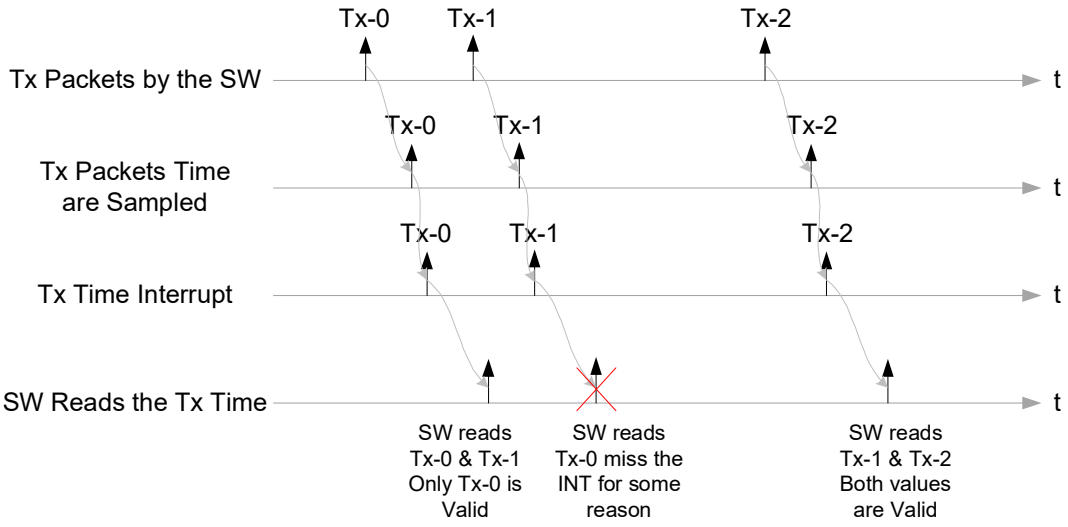


Figure 9-21. Sampled Timestamp

9.7.5.2 Receive Timestamp

The E810 samples the reception time of all packets in the PHY. The sampled time is forwarded from the PHY to the E810 as part of the packet's metadata. It is then posted with the packets in the receive descriptor if enabled in the queue context by the relevant QRXFLXP_CNTXT.*TS* flag. This option is valid only when using 32-byte descriptors.

9.7.6 Synchronized Auxiliary Events

The E810 supports the following global auxiliary events:

- Synchronized events to global I/O signals are described in Section 9.7.6.1.
- Synchronized events to PCI slave access are described in Section 9.7.7.1.

9.7.6.1 Auxiliary 1588 I/O Signals

The E810 supports a total of four single-ended GPIO signals (SDP[20:23]) plus one differential GPIO signal (CLK_OUT_P/N), which is configured by default as 1PPS (out). The functionality of these GPIOs is controlled by the GLGEN_GPIO_CTL[x] registers (expected to be loaded from the NVM). These GPIO pins are associated to the 1588 logic by the *PIN_FUNC* fields in the GLGEN_GPIO_CTL[x] registers.

The GPIO signals are set to input or output by the *PIN_DIR* field in the GLGEN_GPIO_CTL registers. When set to input signal, the sampling event can be sampled by one of the GLTSYN_EVNT registers (as programmed by the *PIN_FUNC* field). When set to output signal, the output event time is defined by one of the GLTSYN_TGT registers (as programmed by the *PIN_FUNC* field). The following settings should be loaded from the NVM and must not be modified by the software:

- GLGEN_GPIO_CTL[5] - CLK_OUT_P/N (out): *PIN_DIR* must be set to output.

There are several output modes of operation described below. In any of them, the initial state of the GPIO pins can be set by the GLGEN_GPIO_CTL registers. The output modes related to the 1588 timers are enabled by the *OUT_ENA* flag in the GLTSYN_AUX registers.

Note: Though the system configuration allows outputting high-frequency signals using the single-ended GPIO signals, the maximum, output frequency supported by the system for these GPIO signals is 20 MHz with a minimum duty cycle of 10%.

- **Synchronized Level Output** — Software should set the event time in the matched GLTSYN_TGT registers. Then set the *OUT_ENA* flag to one and the *OUTMOD* field to "Output Level Mode" (00b) in the GLTSYN_AUX_OUT registers. When the upper 64 bits of the GLTSYN_TIME crosses the value of the GLTSYN_TGT, the assigned GPIO transits to the programmed output level.

  Note: The output level is set by the TimeSync logic only once (when reaching or passing the time specified in GLTSYN_TGT.

- **Synchronized Flipped Output Signal** — Same flow as the Synchronized Level Output, except set the *OUTMOD* field to "Flipped Output Mode" (01b) in the GLTSYN_AUX_OUT registers. When the upper 64 bits of the GLTSYN_TIME crosses the value of the GLTSYN_TGT the GPIO flips its output level.

  Note: The output is flipped by the TimeSync logic only once (when reaching or passing the time specified in GLTSYN_TGT.

- **Synchronized Output Pulse** — Same flow as the Synchronized Level Output, except set the *OUTMOD* field to "Output Pulse Mode" (10b) and the *PULSEW* field to the required pulse width in the GLTSYN_AUX_OUT registers. Doing so, when the upper 64 bits of the GLTSYN_TIME crosses the value of the GLTSYN_TGT, the GPIO signal flips its output state for {16 x (*PULSEW* + 1)} x 1588 clocks and then reverts back to its previous level.

  Note: The output is pulsed by the TimeSync logic only once (when reaching or passing the time specified in GLTSYN_TGT.

- **Synchronized Clock Output** — This mode is selected by setting the *OUT_ENA* flag to one and the *OUTMOD* field to "Output Clock Mode" (11b) in the GLTSYN_AUX_OUT register. The matched GLTSYN_CLKO registers should be set to 50% of the required output clock duration, and the GLTSYN_TGT registers should be set to the time on which the output clock should start. Each time the upper 64 bits of the GLTSYN_TIME crosses the value of the GLTSYN_TGT the GPIO signal flips its output level. Then, the GLTSYN_TGT register is reloaded by the hardware to GLTSYN_TGT plus GLTSYN_CLKO (while GLTSYN_CLKO is padded by 32 MS bit zero's).

  Note: For proper operation, the GLTSYN_CLKO must be larger than twice the value of GLTSYN_INCVAL, and for reasonable accuracy, GLTSYN_CLKO should be significantly larger than twice the value of GLTSYN_INCVAL.

  Software can set the initial state of the output signal by direct setting of the GPIO level.

- **Sampling Input Event** — The E810 can capture the time on which a level transition is sensed on the 1588 auxiliary signals. The event time is captured by one of the 64 bit GLTSYN_EVNT registers. The sampled input event type is defined by the *EVNTLVL* field in the GLTSYN_AUX_IN registers, equal to one of the following options:

  — Disable (00b)

  — Rising Edge (01b)

  — Falling Edge (10b)

  — Any Transition (11b)

  When the defined transition is sampled by the hardware (synchronized to the 1588 clock), the upper 64 bits of the GLTSYN_TIME are latched by the matched GLTSYN_EVNT registers.

9.7.7 Synchronization with Host Timer

The time relationship between the CPU time and devices within the system is a key element in many applications. Specifically, the relationship with the 1588 clock is critical. The following subsections describe mechanisms to measure this relationship.

9.7.7.1 Reading and Sampling the 1588 Master Timers

Reading and sampling the 1588 master timers is described in Section 9.7.4.6. Software can use the "sample time" flow described in Section 9.7.4.6 to get the time delta between the two 1588 master timers.

9.7.8 Interrupts

The E810 can generate a 1588 interrupt for one of the following events if the interrupt is enabled by the *TSYN* flags in the PFINT_OICR_ENA register.

- The PHY asserts an interrupt signal to the E810 for each transmit packet that is time sampled. Packet on port 'n' asserts bit 'n' in the GLINT_TSYN_PHY register. The PF that owns the port as programmed by the PFINT_TSYN_MSK register gets an interrupt if enabled by the *TSYN_TX* flag.
- Input event is latched by one of the GLTSYN_EVNT registers while the interrupt is enabled by the *TSYN_EVNT* flag. The interrupts is asserted to the PF defined by the *PF_MASTER* field in the GLINT_TSYN_PFMSTR0 and GLINT_TSYN_PFMSTR1 registers for an event that is generated by the 1588 master timer 0 and 1, respectively.

- One of the target time registers has expired while the interrupt is enabled by the *TSYN_TGT* flag. See above for PF *PF_MASTER* impact.

Regardless if the interrupt is enabled, the above AUX events are reported in the applicable PFINT_OICR register.

9.7.9 1588 Initialization Flow

This section describes the PF software initialization flow required to activate the 1588 logic.

1. Check the *PF_MASTER* field in the GLINT_TSYN_PFMSTR register if the PF is expected to control the 1588 logic.
2. The 1588 CSRs in the PHYs are initialized by global reset, which might not be triggered by a PCI reset. Therefore, PF software is required to initialize most of the 1588 registers listed in the "TimeSync (IEEE 1588) Registers" section:
   a. Clear any optional residuals reported in the GLTSYN_STAT register and the GLINT_TSYN_PHY register.
   b. Read all transmit timestamps in the PHYs clearing its possible valid indication.
   c. Set the GLTSYN_AUX register as required for the AUX functionality.
   d. The timer and the increment values are expected to be programmed as part of the nominal operation (GLTSYN_TIME and GLTSYN_INCVAL registers).
3. Assign a transmit queue for 1588 transmission and enable *TSYN_ENA* flag in its queue context.
4. Set the relevant QRXFLXP_CNTXT.*TS* flag in those receive queues that should get the receive timestamps.

9.7.10 Software Timer

The E810 includes a 32-bit counter in GLVFGEN_TIMER register, which is based on a free running 1 μs clock. The counter is cleared by Power On Reset (POR) and increments after being cleared. The timer wraps around in about 70 minutes.

9.8 LLDP Protocol

9.8.1 Introduction

The E810 supports the IEEE Data Center Bridging standards such as:

- IEEE 802.1Qaz — Enhanced Transmission Selection (ETS)
- IEEE 802.1Qbb — Priority based Flow Control (PFC)
- IEEE 802.1Qbg — Edge Virtual Bridging (ECB)

Devices that support these standards use IEEE 802.1AB Link Layer Discovery Protocol (LLDP) to exchange configuration information with their network link partner.

LLDP is a link layer protocol that allows a LAN station to advertise capabilities and status of the system. An LLDP agent transmits and receives information to and from the LLDP agents of other stations attached to the same LAN. The information distributed and received in each LLDP Data Unit (LLDPDU) is stored in two Management Information Bases (MIBs) per physical LAN port, one for Nearest Bridge and the other for non-TPMR.

9.8.2 Scope

The E810 supports an embedded LLDP agent that runs on the Embedded Management Processor (EMP).

This section describes implementation of the embedded LLDP agent. It describes the agents operational modes, supported TLVs, configuration, and run time operation of LLDP.

9.8.3 LLDP Agent

The following IEEE standards have configuration parameters and use LLDP to exchange configuration parameters with the link partner.

- Data Center Bridging (DCB).

The DCB features (PFC, ETS, DCBX, App TLV) are defined as requirements for a VLAN aware bridge component. Both C-components and S-components are VLAN-aware bridge components. The E810 supports a single instance of DCB logic per physical LAN port. Therefore, the E810 DCB logic is associated with the component connected to the physical LAN port, either C or S depending on the use case/configuration.

The E810 supports an LLDP agent that exchanges LLDP MIB with its peer. The LLDP agent exchanges the LLDP MIB with either an S-VLAN bridge or C-VLAN bridge depending on mode of operation.

The LLDP agent is active under the following conditions

- During pre-boot operations, including S5 (D0u and D0a).
- During OS present mode, unless software explicitly turns off the agent using the Stop LLDP Agent AQ command.

The LLDP agent is enabled during power-on by setting the LLDP Admin Status word in NVM, (enabled separately per Ethernet port). It is disabled when the Stop LLDP Agent AQ command is executed (per Ethernet port). Software can transfer ownership of LLDP processing back to the device by issuing a Start LLDP Agent AQ command.

9.8.4 LLDP Processing

9.8.4.1 LLDPDU Addressing and Forwarding

9.8.4.1.1 Egress Rules

On the egress side, the LLDP agent uses the appropriate group MAC Address as destination MAC and with EtherType 0x88CC:

- **DCB** — Nearest bridge address.

When an LLDP agent is enabled in the device, the following forwarding rules apply for untagged LLDP packet originating in the host:

- Packets with a Nearest Bridge destination MAC Address are forwarded to the internal control port.
- Packets with Nearest Customer Bridge destination MAC Address are dropped.
- Packets with a non-TPMR destination MAC Address are forwarded to the internal control port.

A control port that wants to send control packets overriding these rules should use the *SWTCH* field in the transmit context descriptor of the control packet. Tagged (802.1Qbg) LLDP packets are forwarded like regular packets. The driver can define different rules (forward to control VSI or drop) using the Add Switch Rules admin command (Section 7.8.12.6.1).

9.8.4.1.2 Ingress Rules

When an LLDP agent is enabled in the device, the following forwarding rules apply for Rx LLDP packets:

Table 9-70. Forwarding Rules for Rx LLDP Packets

Destination MAC Address	STagged LLDP Packets	Untagged LLDP Packets
Nearest Bridge Address	Forward to the PF if programmed so by the PF. Drop otherwise	Forward to EMP.
Non-TPMR Address	Forward to the PF if programmed so by the PF. Drop otherwise	Forward to EMP.
Nearest Customer Bridge Address	Forward to the PF if programmed so by the PF. Drop otherwise	Forward to the PF if programmed so by the PF. Drop otherwise

9.8.4.2 Supported TLV

The LLDP agent should include following TLV as part of the LLDPDU.

- **Chassis ID TLV** — One of four mandatory TLVs. Uses *TLV Type* value of 1 and subtype value of 4 (MAC Address). This TLV contains the permanent/factory-assigned MAC Address of the LAN port.
- **Port ID TLV** — One of four mandatory TLVs. Uses *TLV Type* value of 2 and subtype value of 3 (MAC Address). This TLV contains MAC Address of any physical function. If there is none assigned to a physical function, this TLV contains the permanent/factory-assigned MAC Address of the LAN port.
- **TTL TLV** — One of four mandatory TLVs. Uses *TLV Type* value 3. This TLV contains time-to-live value and is the lower between ((*msgTxHold* * *msgTxInterval*) +1) and 65535. Default *msgTxHold* and *msgTxInterval* are defined in NVM and are loaded by the agent during initialization. See Section 9.8.5.2 for default values.
- **End of LLDPDU TLV** — One of four mandatory TLVs. Uses *TLV Type* value 0. This TLV does not contain any information and sets the TLV information string length to 0.

  Note: Although this is a mandatory TLV, there are apparently some implementations out there that do not send it. Therefore, the device does not require that an LLDPDU ends with this TLV.
- **OEM Device type TLV** — One of the three OEM required TLVs. This TLV uses *TLV Type* value of 127 with subtype of 1. For more information, refer to OEM-specific LLDP specifications for OUI and contents of the TLV.
- **OEM Firmware Version TLV** — One of the three OEM required TLVs. This TLV uses *TLV Type* value of 127 with subtype of 3. For more information, refer to OEM-specific LLDP specifications for OUI and contents of the TLV.
- **OEM Port Capabilities TLV** — One of the three OEM required TLVs. This TLV uses *TLV Type* value of 127 with subtype of 4. For more information, refer to OEM-specific LLDP specifications for OUI and contents of the TLV.
- **DCBX ETS Configuration TLV** — This TLV uses *TLV Type* value of 127 with OUI of IEEE 802.1, which is 0x0080C2. Subtype is 0x09.

  This TLV information string uses the following default values:

  — *Willing* bit set to 1 (unless the software decides to take control by calling Set Local MIB and setting the *willing* bit to 0), indicating that this station is willing to accept configuration from remote station.

  — *CBS* bit is set to 0, indicating that this station does not support credit based shaper.

  — Maximum traffic classes.

  — *Priority Assignment Table* is a four octet string with four bits per entry.

  — *Bandwidth Table* is a string of eight octets with each octet defining the bandwidth percentage for traffic classes. Octet 0 specifies bandwidth percentage of Traffic Class 0, octet 1 for Traffic Class 1, and so on.

  — *TSA Assignment Table* is a string of eight octets with an 8-bit value specifying Transmission selection algorithm per traffic class.

- **DCBX PFC Configuration TLV** — This TLV uses type value of 127 with OUI of IEEE 802.1, which is 0x0080C2. Subtype is 0x0B.

  This TLV information string uses the following default values:

  — *Willing* bit set to 1 (unless the software decides to take control by calling Set Local MIB and setting the *willing* bit to 0), indicating that this station is willing to accept configuration from remote station.

  — *MBC* bit is set to 0 since MACsec is not processed by the E810.

  — *PFC Capability* is a 4-bit unsigned integer indicating the number of traffic classes that simultaneously support PFC. Default value is 8.

  — *PFC Enable* is a 8-bit vector that indicates which traffic classes have PFC enabled. Default value is zero. None of the traffic call uses have PFC enabled by default.

- **DCBX Application Priority Configuration TLV** — This TLV uses type value of 127 with OUI of IEEE 802.1, which is 0x0080C2. Subtype is 0x0C.

  By default, this optional TLV is not transmitted by the LLDP agent, but the agent reflects the table received from peer network port if the agent receives LLDPDU with remote link partner.

9.8.4.3 LLDPDU Transmission and Reception

First LLDP transmission is immediate after the LLDP agent reaches ready state and link is up. This first transmission populates the LLDPDU with default TLV values.

The LLDP agent enters Fast transmission state/period whenever the LLDP agent detects a new neighbor. The LLDP agent enters new neighbor detected state (Fast transmission period) when an IEEE 802.1az Link Layer LLDP frame with new Chassis and Port ID is received. Fast transmission state exited after transmitting *txFastInit* number of LLDPDUs. Default value of *txFastInit* is 4.

When not in Fast transmission mode, the LLDP agent transmits every *msgTxInterval*, unless there is change in local LLDP MIB variables. LLDPDU is transmitted immediately, without waiting for *msgTxInterval* when there is a change in local MIB variables.

The E810 supports a single LLDP neighbor per supported destination LLDP address per port. (that is, two total neighbors – one for Nearest Bridge and one for Nearest non-TPMR per port). LLDPDUs with other addresses are ignored.

The transmission and reception flow is interrupted by several events, each causing a disruption in the normal flow. The behavior in such cases is as follows:

- LLDP events visible to both ends:
  — Aging of the LLDP timer or LLDPDU is received with a TTL value of zero (for example, Shutdown LLDPDU).
    - Delete all information in the LLDP remote systems MIB associated with the respective MSAP identifier.
    - EMP reconfigures device based on the local defaults.
  — Missing TLV in a received LLDPDU.
    - Delete all information for the missing TLV in the LLDP remote systems MIB associated with the respective MSAP identifier.
    - EMP reconfigures device based on the local defaults of this TLV.

- Events that cause a link down:
  - Link Down event (only if LLDP timer has not expired).
    - When link is back up, delete all information in the remote systems MIB associated with the LLDP agent(s) for this link.
    - EMP reconfigures device based on the local defaults.
  - Device Resets that causes Link Down (in other words, EMP Reset, Global Reset, PCI Resets that causes Link Down)
    - Device performs an Internal GLOBR that brings the link down momentarily.
    - Continue as in Link Down case.
  - LLDP ownership transfer from software to firmware.
    - Software issues a Global Reset to the device.
    - Software sends a Start LLDP Agent AQ command to start LLDP agent in firmware.
    - Continue as in Link Down case.
- Events not visible to other end:
  - LLDP ownership transfer from firmware to software.
    - LLDP agent terminated in device.
    - No automatic change in device configuration.
    - EMP waits for software commands.
  - Device resets that do not cause Link Down (like Core Reset, PCI Resets that do not cause link down, function-level resets).
    - Device performs an internal CORER.
    - EMP reconfigures core based on the LLDP MIBs.

      Note: No change in MAC/PHY state, including Flow Control configuration and operation.

      LLDP reception and transmission continues once the above configuration is done. Configuration is estimated to take milliseconds, well below the LLDP timeout values (usually in seconds).

9.8.4.4 LLDP Protocol Variables

Unless otherwise specified, the LLDP agent operational state variables are set to values recommended in Section 9.2.2 of the IEEE 802. AB - 2009 standard. However, certain values can be configured via NVM, and are loaded from NVM when the LLDP agent reaches operational state; this occurs after a power-up or a device reset. Refer to Section 9.8.5.2 for a list of LLDP protocol variables that can be configured via NVM.

9.8.4.5 LLDP Data Store

The LLDP agent maintains sufficient information to enable a host software based management agent to support basic LLDP MIB and organizationally-specific LLDP MIB extensions.

The following basic variables are maintained by the LLDP agent:

- ***msgTxInterval*** — Defines the time interval in timer ticks between transmissions during normal transmission periods.
- ***msgTxHold*** — Used, as a multiplier of *msgTxInterval*, to determine the value of txTTL that is carried in LLDP frames transmitted by the LLDP agent.
- ***reinitDelay*** — Indicates the amount of delay from when *adminStatus* becomes "disabled" until re-initialization is attempted.
- ***txCreditMax*** — Maximum number of consecutive LLDPDUs that can be transmitted at any time.
- ***msgFastTx*** — Defines the time interval in timer ticks between transmissions during Fast transmission periods.
- ***txFastInit*** — Determines the number of LLDPDUs that are transmitted during a Fast transmission period.
- ***adminStatus*** — Indicates whether or not the LLDP agent is enabled. The defined values for this variable are as follows:
  - Integer value of 3 means the LLDP agent is enabled for reception and transmission of LLDPDUs.
  - Integer value of 2 means the LLDP agent is enabled for transmission of LLDPDUs only.
  - Integer value of 1 means the LLDP agent is enabled for reception of LLDPDUs only.
  - Integer value of 0 means the LLDP agent is disabled for both reception and transmission.

The LLDP agent maintains last received LLDPDU per port. Upper bound for size of LLDPDU is 1500 bytes. Details of data structure used for information store is implementation specific and beyond scope of this document.

The E810 provides LSAP services to the OS provided MSAP services that are used by LLDP agents hosted by system processors.

9.8.5 Initialization and Configuration

9.8.5.1 Initialization

As part of EMP initialization, an the LLDP agent is loaded and initialized by EMP. The LLDP agent loads default values for TLVs from the NVM. The LLDP agent transitions to "ready" state and waits for LAN link up before transmitting the first LLDPDU.

DCBX agent operates in slave mode, sets the *Willing* bit, and accepts recommendations from the link partner (unless the software decides to take control by calling Set Local MIB and setting *willing* bit to 0).

9.8.5.2 LLDP Configuration

9.8.5.2.1 LLDP Protocol Variables

The following LLDP timers can be configured via NVM:

- ***msgFastTx*** — Defines the time interval in timer ticks between transmissions during Fast transmission periods. The default value of *msgFastTx* is 1. This value can be changed by management to any value in the range 1 through 3600.
- ***msgTxInterval*** — Defines the time interval in timer ticks between transmissions during normal transmission. The default value for *msgTxInterval* is 30 seconds. This value can be changed by management to any value in the range 1 through 3600.
- ***msgTxHold*** — Used as a multiplier of *msgTxInterval*, to determine the value of txTTL that is carried in LLDP frames transmitted by the LLDP agent. The recommended default value of msgTxHold is 4. This value can be changed by management to any value in the range 1 through 100.
- ***txCreditMax*** — Determines maximum number of LLDPDUs that can be sent per second. The default value of *txCreditMax* is 5. This value can be changed by management to any value in the range 1 through 10.
- ***txFastInit*** — Determines the number of LLDPDUs that are transmitted during a Fast transmission period. The default value of *txFastInit* is 4. This value can be changed by management to any value in the range 1 through 8.
- ***reinitDelay*** — Indicates the amount of delay from when *adminStatus* becomes "disabled" until re-initialization is attempted. Default value of *reinitDelay* is 2.

9.8.5.2.2 LLDP Admin Queue Commands

The following commands are supported by the E810 to manage the LLDP agent and provide information to the drivers.

Table 9-71. LLDP Admin Queue Commands

Command	Opcode	Type	Description	Section Reference
Get LLDP MIB	0x0A00	Indirect	Fetch latest LLDP MIB.	9.8.5.2.2.1
Configure LLDP MIB Change Event	0x0A01	Direct	Request and deliver a notification that the peer has sent an updated LLDP MIB.	9.8.5.2.2.2
Add LLDP TLV	0x0A02	Indirect	Add a new TLV to the local LLDP MIB.	9.8.5.2.2.3
Update LLDP TLV	0x0A03	Indirect	Update an existing TLV in the local LLDP MIB.	9.8.5.2.2.4
Delete LLDP TLV	0x0A04	Indirect	Delete a TLV from the local LLDP MIB.	9.8.5.2.2.5
Stop LLDP Agent	0x0A05	Direct	Used to stop or shutdown LLDP agent.	9.8.5.2.2.6
Start LLDP Agent	0x0A06	Direct	Start an LLDP agent running.	9.8.5.2.2.7
Get CEE DCBX CFG	0x0A07	Indirect	Retrieves the CEE configuration.	9.8.5.2.2.8
Set Local LLDP MIB	0x0A08	Indirect	Load the DCBX configuration.	9.8.5.2.2.9
Stop/Start a Specific LLDP Agent	0x0A09	Direct	Stop and restart the firmware DCBX agent.	9.8.5.2.2.10
LLDP Filter Control	0x0A0A	Direct	Request forwarding of LLDP traffic to host.	9.8.5.2.2.11

9.8.5.2.2.1 Get LLDP MIB (0x0A00)

This command, posted on the ATQ, is an indirect AQ command. The driver requests the complete LLDP MIB, providing a response buffer (address/length pair) and the type of LLDP MIB requested. The LLDP MIB is associated with a physical LAN port. Instead of formatting the LLDP MIB in any particular way, firmware should write the entire packet (including headers) that was sent/received on the wire for the particular MIB type specified. The MIB is guaranteed to fit in a 1.5K packet, so there is no need to use a "large buffer". For a particular Bridge Type, software can request the local MIB, the remote MIB, or both MIBs.

Firmware writes back the complete LLDP MIB to the response buffer. It writes the length of the LLDP MIB into the *Datalen* field in the descriptor on the ATQ. It writes the status of the request in the *Return Value* field. For the case where both MIBs are returned, firmware always writes the Local MIB first and the Remote MIB immediately following the Local MIB.

Table 9-72. Get LLDP MIB Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0A00	Command opcode.
Datalen	4-5		Length or response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Type	16	MIB Type	**Bits 0:1:** Direction 00b = Local MIB (sent by device) 01b = Remote MIB (received by device) 10b = Both Local and Remote MIBs 11b = Reserved **Bits 2:3:** Bridge Type 00b = Nearest Bridge 01b = Non-TPMR Bridge 10b = Reserved 11b = Reserved **Bits 4:7:** Reserved
Reserved.	17-23	Reserved	Reserved.
Data Address High	24-27		Address of response buffer.
Data Address Low	28-31		

Table 9-73. Get LLDP MIB Response

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0A00	Command opcode.
Datalen	4-5		Length of LLDP MIB if Status==Success. In bytes.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. Status of request. A value of SUCCESS means that command was performed successfully. Error Codes: ENOENT = Firmware returns this value if any requested LLDP MIB does not exist. EPERM = Firmware returns this value if software has taken control of LLDP processing. EFBIG = Firmware returns this value when size of LLDPDU is larger than size of the response buffer. EINVAL = Firmware returns this value when software asks for a bad request (for example, invalid bridge type or direction).
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Type	16	MIB Type	**Bits 0:1:** Direction 00b = Local MIB (sent by device) 01b = Remote MIB (received by device) 10b = Both Local and Remote MIBs 11b = Reserved **Bits 2:3:** Bridge Type 00b = Nearest Bridge 01b = Non-TPMR Bridge 10b = Reserved 11b = Reserved **Bits 4:7:** Reserved
Reserved	17	Reserved	Reserved.
Local MIB Length	18-19	Length	If the response buffer contains a Local MIB, this field reports its length. If no Local MIB is present, this field is written with a value of 0.
Remote MIB Length	20-21	Length	If the response buffer contains a Remote MIB, this field reports its length. If no Remote MIB is present, this field is written with a value of 0.
Reserved	22-23	Reserved	Reserved.
Data Address High	24-27		Address of response buffer.
Data Address Low	28-31		

Note: If only one MIB is present, it is written to the response buffer beginning at the first byte of the response buffer (i.e. offset=0). If two MIBs are present, the Local MIB is always written first (i.e. offset=0), and the Remote MIB is written immediately following the Local MIB (i.e. offset=*LocalMIBLength*).

9.8.5.2.2.2 Configure LLDP MIB Change Event (0x0A01)

This command, posted on the ATQ, is a direct AQ command. The driver uses this command to request that firmware post an event on the ARQ when the LLDP MIB associated with this interface changes. The driver can also use this command to request that firmware stop posting this event to the ARQ.

Firmware writes back the status of the request to the *Return Value* field.

Table 9-74. Configure LLDP MIB Change Event Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0A01	Command opcode.
Datalen	4-5	0	Direct command. No response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Command	16	Command	**Bit 0:** Command 0b = Enable event 1b = Disable event **Bits 1:7:** Reserved
Reserved	17-31	Reserved	Reserved.

Table 9-75. Configure LLDP MIB Change Event Response

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0A01	Command opcode.
Datalen	4-5	0	Direct command. No response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. Status of request. A value of SUCCESS means that command was performed successfully. Error Codes: EPERM = Firmware returns this value if software has taken control of LLDP processing.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-31	Reserved	Reserved.

9.8.5.2.2.2.1 LLDP MIB Change Event

This event is posted on the ARQ to indicate to software that any LLDP MIB associated with the physical interface has changed. The LLDP MIB change event is also posted if a multiple-peers condition is detected or if a TLV is aged out.

Firmware provides the status of the event, the length of the LLDP MIB in bytes, and the type of LLDP MIB that has changed, and copies the *Cookie* value from the associated Configure LLDP MIB Change Event.

The response buffer includes the entire MIB that has changed. The formatting is identical to the response for the Get LLDP MIB command.

Note: The opcode for this event is the same as the opcode for the related command on the ATQ that enabled this event to be sent to software.

Table 9-76. LLDP MIB Change Event

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0A01	Event code.
Datalen	4-5	0	Length of LLDP MIB if Status==Success. In bytes.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. Status of request.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Type	16	MIB Type	**Bits 0:1:** Direction 00b = Local MIB (sent by device) 01b = Remote MIB (received by device) 10b = Reserved 11b = Reserved **Bits 2:3:** Bridge Type 00b = Nearest Bridge 01b = Non-TPMR Bridge 10b = Reserved 11b = Reserved **Bits 4:5:** Miscellaneous 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. **Bits 6:7:** Reserved
Reserved	17-23	Reserved	Reserved.
Data Address High	24-27		Address of response buffer.
Data Address Low	28-31		

9.8.5.2.2.3 Add LLDP TLV (0x0A02)

This command, posted on the ATQ, is an indirect AQ command. The driver provides the type of MIB to be updated, the TLV to be added, and the address/length of the buffer containing the TLV. Software is responsible for guaranteeing that the TLV to be added does not already exist in the MIB, or follows the TLV usage rules for TLVs that allow multiple instances. Software can only add TLVs to the Local LLDP MIB.

Firmware adds the new TLV to the Local LLDP MIB just before the "End of LLDPDU TLV". Firmware writes back the complete LLDP MIB to the response buffer. It writes the length of the LLDP MIB into the *Datalen* field in the descriptor on the ATQ.

Table 9-77. Add LLDP TLV Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0A02	Command opcode.
Datalen	4-5	Length	Length of indirect buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Type	16	MIB Type	**Bits 0:1:** Reserved **Bits 2:3:** Bridge Type 00b = Nearest Bridge 01b = Non-TPMR Bridge 10b = Reserved 11b = Reserved **Bits 4:7:** Reserved
Reserved	17	Reserved	Reserved.
Length	18-19	Length	Length of TLV placed in the indirect buffer by the driver.
Reserved	20-23	Reserved	Reserved.
Data Address High	24-27		Address of indirect buffer.
Data Address Low	28-31		

Table 9-78. Add LLDP TLV Response

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0A02	Command opcode.
Datalen	4-5		Length of LLDP MIB if Status==Success. In bytes.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. Status of request. Error Codes: ENOMEM = Firmware returns this value if there is not enough space available to add the new TLV. EINVAL = Firmware returns this value if the MIB Type associated with this command does not exist. EPERM = Firmware returns this value if software has taken control of LLDP processing.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.

Table 9-78. Add LLDP TLV Response [continued]

Name	Byte.Bit	Value	Remarks
Type	16	MIB Type	**Bits 0:1:** Reserved **Bits 2:3:** Bridge Type 00b = Nearest Bridge 01b = Non-TPMR Bridge 10b = Reserved 11b = Reserved **Bits 4:7:** Reserved
Reserved	17-23	Reserved	Reserved.
Data Address High	24-27		Address of response buffer.
Data Address Low	28-31		

9.8.5.2.2.4 Update LLDP TLV (0x0A03)

This command, posted on the ATQ, is an indirect AQ command. The driver provides the bridge type of the MIB to be updated, the TLV to be updated (both original and updated versions), the offset/length of both TLVs in the indirect buffer, and the address/length of the indirect buffer. Software is responsible for guaranteeing that the TLV to be updated does already exist in the MIB. Only a local MIB can be updated by the driver.

Firmware must match the entire original TLV in the MIB before performing an update. There are some TLV types that can appear more than once. Therefore, matching based only on Type/Subtype is not sufficient.

Firmware writes back the complete LLDP MIB to the response buffer. It writes the length of the LLDP MIB into the *Datalen* field in the descriptor on the ATQ.

Table 9-79. Update LLDP TLV Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0A03	Command opcode.
Datalen	4-5	Length	Length of indirect buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Type	16	MIB Type	**Bits 0:1:** Reserved **Bits 2:3:** Bridge Type 00b = Nearest Bridge 01b = Non-TPMR Bridge 10b = Reserved 11b = Reserved **Bits 4:7:** Reserved
Reserved	17-23	Reserved	Reserved.
Length1	18-19	Length	Length of original TLV in the indirect buffer. Offset is assumed to be 0.
Offset2	20-21	Offset	Offset of updated TLV in the indirect buffer.
Length2	22-23	Length	Length of updated TLV in the indirect buffer.

Table 9-79. Update LLDP TLV Command [continued]

Name	Byte.Bit	Value	Remarks
Data Address High	24-27		Address of indirect buffer.
Data Address Low	28-31		

Table 9-80. Update LLDP TLV Response

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0A03	Command opcode.
Datalen	4-5		Length of LLDP MIB if Status==Success. In bytes.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. Status of request. Error Codes: EINVAL = Firmware returns this value if the requested MIB does not exist. ENOXIO = Firmware returns this value if the "original" TLV does not exist in the requested MIB. ENOMEM = Firmware returns this value if the updated TLV is larger than the original TLV and there is not enough space to increase the size of the TLV. EPERM = Firmware returns this value if software has taken control of LLDP processing.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Type	16	MIB Type	**Bits 0:1:** Reserved **Bits 2:3:** Bridge Type 00b = Nearest Bridge 01b = Non-TPMR Bridge 10b = Reserved 11b = Reserved **Bits 4:7:** Reserved
Reserved	17-23	Reserved	Reserved.
Data Address High	24-27		Address of response buffer.
Data Address Low	28-31		

9.8.5.2.2.5 Delete LLDP TLV (0x0A04)

This command, posted on the ATQ, is an indirect AQ command. The driver provides the type of MIB to be updated and a copy of the TLV to be deleted. Software is responsible for guaranteeing that the TLV to be deleted does already exist in the MIB.

Firmware writes back the complete LLDP MIB to the response buffer. It writes the length of the LLDP MIB into the *Datalen* field in the descriptor on the ATQ.

Table 9-81. Delete LLDP TLV Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0A04	Command opcode.
Datalen	4-5	Length	Length of indirect buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Type	16	MIB Type	**Bits 0:1:** Reserved **Bits 2:3:** Bridge Type 00b = Nearest Bridge 01b = Non-TPMR Bridge 10b = Reserved 11b = Reserved **Bits 4:7:** Reserved
Reserved	17	Reserved	Reserved.
Length	18-19	Length	Length of TLV to be deleted. The TLV itself is copied into the indirect buffer by the driver.
Reserved	20-23	Reserved	Reserved.
Data Address High	24-27		Address of indirect buffer.
Data Address Low	28-31		

Table 9-82. Delete LLDP TLV Response

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0A04	Command opcode.
Datalen	4-5		Length of LLDP MIB if Status==Success. In bytes.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. Status of request. Error Codes: EINVAL = Firmware returns this value if the requested MIB does not exist. ENOXIO = Firmware returns this value if the TLV to be deleted does not exist in the requested MIB. EPERM = Firmware returns this value if software has taken control of LLDP processing.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.

Table 9-82. Delete LLDP TLV Response [continued]

Name	Byte.Bit	Value	Remarks
Type	16	MIB Type	**Bits 0:1:** Reserved **Bits 2:3:** Bridge Type 00b = Nearest Bridge 01b = Non-TPMR Bridge 10b = Reserved 11b = Reserved **Bits 4:7:** Reserved
Reserved	17-23	Reserved	Reserved.
Data Address High	24-27		Address of response buffer.
Data Address Low	28-31		

9.8.5.2.2.6 Stop LLDP Agent (0x0A05)

This command, posted on the ATQ, is a direct AQ command. The driver uses this command to request that firmware stop or shutdown the LLDP agent on the port.

If *Stop* is specified, the device stops the LLDP agent on the port and directs all untagged ingress LLDP frames received on the port to the default queue of the Control VSI associated with the Port aggregator or Port extender.

If *Shutdown* is specified, the device stops the LLDP agent on the port and sends a last LLDP PDU on the wire with TTL=0 and with Nearest Bridge and non-TPMR destination address. Firmware then directs all untagged ingress LLDP frames received on the port to the default queue of the S-Component Control VSI.

When *Command* Bit 1 is set to 1b to indicate disabling LLDP Agent persistently, it sets the *Current LLDP AdminStatus* for the given Port to 0x0. EMP firmware does not attempt to stop the LLDP Agent on that port if the agent is already disabled.

Firmware writes back the status of the request.

Any preceding registration to events on the port (via the Configure LLDP MIB Change Event command) is discarded. Software should register for events again once the LLDP agent in the device is active again.

After the response, the driver should route the LLDP flows to a control VSI using the Add Switch Rules admin command (Section 7.8.12.6.1).

Table 9-83. Stop LLDP Agent Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0A05	Command opcode.
Datalen	4-5	0	Direct command. No response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.

Table 9-83. Stop LLDP Agent Command [continued]

Name	Byte.Bit	Value	Remarks
Command	16	Command	**Bit 0:** Command 0b = Stop LLDP agent 1b = Shutdown LLDP agent **Bit 1:** Command 0b = No effect 1b = Persistent disablement of LLDP Agent **Bits 2:7:** Reserved
Reserved	17-31	Reserved	Reserved.

Table 9-84. Stop LLDP Agent Response

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0A05	Command opcode.
Datalen	4-5	0	Direct command. No response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. Status of request. A value of SUCCESS means that command was performed successfully. Error Codes: EPERM = Firmware returns this value if software has taken control of LLDP processing.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-31	Reserved	Reserved.

9.8.5.2.2.7 Start LLDP Agent (0x0A06)

It is expected that CORER has occurred before this command is issued. CORER causes the EMP to reload LLDP forwarding rules from NVM default or based on the *Current LLDP AdminStatus* and re-initialize based on those settings.

This command, posted on the ATQ, is a direct AQ command. In response to this command EMP re-enables LLDP agent over a given port and treats the command as a request to set the LLDP Configuration variable *AdminStatus* that indicates the LLDP Agent is enabled for "Both Tx and Rx enabled". EMP firmware starts the firmware-based LLDP Agent regardless of the NVM settings in either default *LLDP Admin Status* or the *Current LLDP AdminStatus* fields.

When *Command* Bit 1 is set to 1b to indicate disabling LLDP Agent persistently, it sets the *Current LLDP AdminStatus* for the given Port to 0x3 indicating the LLDP Configuration AdminStatus is set to "Both Tx and Rx enabled" mode. EMP firmware does not attempt to stop the LLDP Agent on that port if the agent is already disabled.

If the driver defined LLDP forwarding rules previous to sending this command, these rules should be removed using the Remove Switch Rules admin command (Section 7.8.12.6.3) before sending this command.

Table 9-85. Start LLDP Agent Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0A06	Command opcode.
Datalen	4-5	0	Direct command. No response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Command	16	Command	**Bit 0:** Command 0b = Do not start the LLDP agent **(Note: This value should not be used)** 1b = Start the LLDP agent **Bit 1**: Command 0b = No effect 1b = Persistent disablement of LLDP Agent **Bits 2:7:** Reserved
Reserved	17-31	Reserved	Reserved.

Table 9-86. Start LLDP Agent Response

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0A06	Command opcode.
Datalen	4-5	0	Direct command. No response buffer.
Return Value/VFID	6-7		Return value. The following error values can be returned: EEXIST = LLDP agent is already running in firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-31	Reserved	Reserved.

9.8.5.2.2.8 Get CEE DCBX OPER CFG (0x0A07)

This command, posted on the ATQ, is an indirect AQ command. The driver requests the operational configuration of CEE/DCBX. The driver provides a response buffer (address/length pair).

EMP writes back the CEE/DCBX configuration to the response buffer.

Table 9-87. Get CEE DCBX OPER CFG Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0A07	Command opcode.
Datalen	4-5		Length of response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-23	Reserved	Reserved.
Data Address High	24-27		Address of response buffer.
Data Address Low	28-31		

Table 9-88. Get CEE DCBX OPER CFG Response

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0A07	Command opcode.
Datalen	4-5		Length of LLDP MIB if Status==Success. In bytes.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. Status of request. A value of SUCCESS means that command was performed successfully. Error Codes: ENOENT = Firmware returns this value if any requested LLDP MIB does not exist. EPERM = Firmware returns this value if software has taken control of LLDP processing.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-23	Reserved	Reserved.
Data Address High	24-27		Address of response buffer.
Data Address Low	28-31		

Table 9-89. Get CEE DCBX OPER CFG Response Buffer Format

Offset (Bytes)	Description
0	**Local Oper Num Traffic Class Supported** Returns a value between 1-8 for the number of TCs supported locally.
1-4	**Local Oper Priority Assignment** For each UP a 4-bit for the TCID. Available value is 0-7. Upper bit == 0 Bits 0:3 assigned for up 0. Bits 4:7 assigned for up 1. . . . Bits 27:31 assigned for up 7.
5-12	**Local Oper TCB Bandwidth** An 8-byte field. Each byte represents the relative bandwidth allocation of one enabled TC. Byte 7 assigned for TC 0. Relative bandwidth allocation is a value 0-100, and represents the percentage of the available bandwidth this TC is allocated with. The sum of the bandwidth allocated for all TC equal to 100%
13	**Local Oper PFC Enable** A bitmap containing a PFC-enable flag for each UP. Bit 0 concerns UP 0.
14-15	**Local Oper Application Priority** Indicates the Application TLV negotiated for FCoE, iSCSI, and FIP (encoded in three bits) Bits 14.0-14.2: FCoE Application priority. Bits 14.3-14.5: Reserved for iSCSI Application priority. Bits 14.6-14.7: Reserved. Bits 15.0-15.2: Reserved for FIP Application priority. Bits 15.3-15.7: Reserved.
16-19	**Status flags for DCBX TLVs** For each TLV, indicates the status of the TLV (1 = True, 0 = False): Bit 0 = Operational Mode. Bit 1 = Synced. Bit 2 = Error Bits 16.0-16.2: Status bits for PG. Bits 16.3-16.5: Status bits for PFC. Bits 16.6-16.7: Reserved. Bits 17.0-17.2: Status bits for FCoE Application Priority. Bits 17.3-17.5: Status bits for iSCSI Application Priority. Bits 17.6-17.7: Reserved. Bits 18.0-18.2: Status bits for FIP Application Priority. Bits 18.3-18.7: Reserved. Bits 19.0-18.7: Reserved.
20-31	Reserved. Return 0.

9.8.5.2.2.9 Set Local LLDP MIB (0x0A08)

This command, posted on the ATQ, is an indirect AQ command. The driver configures the complete DCBX MIB, providing a response buffer (address/length pair). The DCBX MIB is associated with a physical LAN port and is expressed in IEEE format even though firmware finally sends it as CEE TLVs on the wire, if needed. The MIB is guaranteed to fit in a 1.5K packet, so there is no need to use a "large buffer". Once DCBX negotiation with the peer completes, the local DCBX MIB returned by the Get LLDP MIB command reflects the final resolved values. In the future, the command could be extended to support more other LLDP MIBs.

This command is useful to configure the local DCBX agent into the non-willing mode (a.k.a. master mode), and to set the DCB configuration to be pushed to the peer.

Firmware writes back the status of the request in the *Return Value* field.

If the *Willing* bit is set in a DCBX TLV, the DCBX agent should take it in account when resolving DCBX with the peer, as per the rules defined in the IEEE/CEE standard.

Firmware can change the local configuration twice, once upon reception of the AQ command to align default configuration to what is published in the TLVs sent to the peer, and once upon reception of the peer's TLV when resolving DCBX. These two steps can be collapsed into one single configuration change in case peer's TLV is received within short delays. The new default configuration is maintained until the next GLOBR event or until a Stop DCBX Agent AQ command is received.

If the command is received while the firmware DCBX agent is disabled or stopped, the MIB is parsed by firmware and used to configure the local DCB settings of the port, with no DCB TLV exchange with the peer performed by firmware. Firmware drains the Tx-Pipe if TC or PFC changes were pushed, as if it was resulting from a regular DCBX negotiation flow.

Table 9-90. Set Local LLDP MIB Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0A08	Command opcode.
Datalen	4-5		Length of response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Type	16	MIB Type	**Bit 0:** 0b = Local DCBX MIB (sent by device). **Bit 1:** 0b = CEE DCB, APP TLV operates in willing mode. 1b = CEE DCB, APP TLV operates in non willing mode. **Bits 2:7:** Reserved
Reserved	17	Reserved	Reserved.
Local MID Length	18-19	Length	Length of the command buffer.
Reserved	20-23	Reserved	Reserved.
Data Address High	24-27		Address of command buffer.
Data Address Low	28-31		

Table 9-91. Set Local LLDP MIB Response

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0A08	Command opcode.
Datalen	4-5		Length of LLDP MIB if Status==Success. In bytes.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. Status of request. A value of SUCCESS means that command was performed successfully. Error Codes: EPERM = If any local LLDP MIB set in the command does not exist (e.g. if the specific agent was stopped). EINVAL = If a DCBX TLV is missing in the pushed MIB or if the TLV's pushed are not consistent or illegal.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Type	16	MIB Type	**Bit 0:** 0b = MIB is Silently Changed. Event to software is not required. 1b = MIB is Changed. MIB change event is triggered. **Bits 1:7:** Reserved
Reserved	17-31	Reserved	Reserved.

9.8.5.2.2.10 Stop/Start a Specific LLDP Agent (0x0A09)

This command, posted on the ATQ, is a direct AQ command. The driver uses this command to request that firmware stop or (re-)start the DCBX agent on the port. In the future, the command could be extended to support more specific agents.

Stopping the DCBX agent on the port means the following:

1. When parsing the LLDP TLVs received from the peer, the LLDP agent skips over all DCBX TLVs.
2. The LLDP agent does not send DCBX TLVs to the peer.
3. Local DCB configuration is returned to the hardware default (single TC, no PFC).
4. Get LLDP MIB still returns the remote DCBX MIBs if such are received from the peer.

(Re-)Starting the DCBX agent on the port means the following:

1. The LLDP agent parses and processes the DCBX TLVs received from the peer.
2. The LLDP agent sends DCBX TLVs to the peer.
3. Local DCB configuration is modified according to the DCBX negotiation with peer.

Firmware writes back the status of the request.

Table 9-92. Stop/Start a Specific LLDP Agent Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0A09	Command opcode.
Datalen	4-5	0	Direct command. No response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Command	16	Command	**Bit 0:** Command 0b = Stop DCBX agent. 1b = (Re-)Start DCBX agent. **Bits 1:7:** Reserved.
Reserved	17-31	Reserved	Reserved.

Table 9-93. Start/Start a Specific LLDP Agent Response

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0A09	Command opcode.
Datalen	4-5	0	Direct command. No response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. Status of request. A value of SUCCESS means that command was performed successfully.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Status	16	Command	**Bit 0:** Status 0b = DCBX agent stopped. 1b = DCBX agent active. **Bits 1:7:** Reserved.
Reserved	17-31	Reserved	Reserved.

9.8.5.2.2.11 LLDP Filter Control

Upon reception of this command, the firmware adds a new filter with the requested VSI as destination.

The following resources are reserved for this command: One forwarding rules and one VSI list per port. These resources are accounted as firmware resources.

Note: A Start LLDP command should be preceded by a LLDP Control command with a "delete" action to remove the filters used by software.

Table 9-94. LLDP Filter Control Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.1 for details.
Opcode	2-3	0x0A0A	Command opcode.
Datalen	4-5	0	Direct command. No response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Command Flag	16	Command	**Bits 1:0:** Command 00b =Add 01b =Delete 10b =Update (change VSI number) 11b =Reserved **Bits 2:7:** Reserved.
Reserved	17	Reserved	Reserved.
VSI	18-19		VSI to send the LLDP traffic to (relevant for Add or Update actions),
Reserved	20-31	Reserved	Reserved.

Table 9-95. LLDP Filter Control Response

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.1.2 for details.
Opcode	2-3	0x0A08	Command opcode.
Datalen	4-5		Direct command. No response buffer.
Return Value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. Status of request. A value of SUCCESS means that command was performed successfully. Error Codes: EPERM = When the command is sent while LLDP is owned by firmware. EACESS = When the requested VSI is not owned by the PF. EINVAL = When there is an attempt to update or remove a non existing filter or add an existing filter.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-31	Reserved	Reserved.

Chapter 10 LAN Engine

10.1 Introduction

This section describes the hardware/software interfaces for transmit and receive LAN functionality. Details include:

- Associated device registers.
- Device descriptor formats.
- Initialization and run-time flows illustrating the use of the registers and descriptors.
- Stateless offloads in Rx and Tx data path.

10.2 Queues Allocation and Management

10.2.1 LAN Receive Queue Allocation

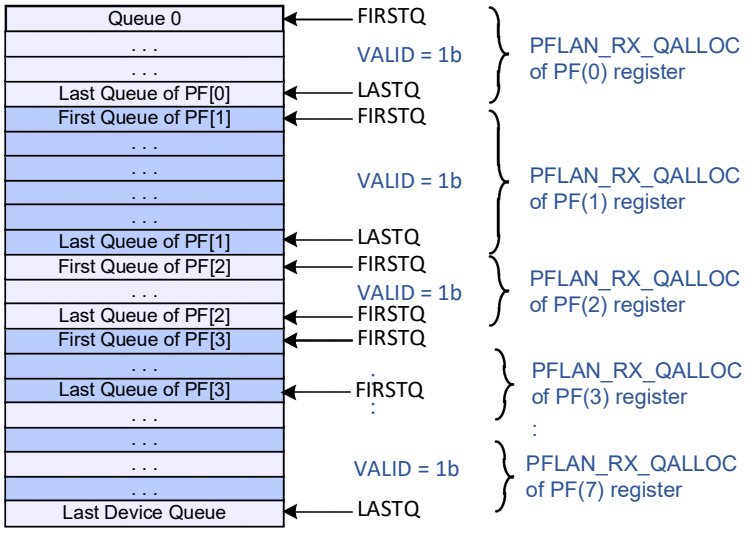


Figure 10-1. LAN Queues Allocation Example of 8 PFs

The PF software device driver is expected to read the values of the PFLAN_RX_QALLOC register (auto-loaded from NVM at reset) to determine the receive queues it owns. The PF allocates these receive queues to VFs by programming the VPLAN_RX_QTABLE and VPLAN_RX_QBASE registers and to VSIs by the Add VSI Admin Queue command, which programs the VSILAN_QTABLE and VSILAN_QBASE registers.

The implementation has the following configuration notes:

- VFs used by the VMM are expected to be statically allocated and therefore could be contiguous. Contiguous space is defined by the *VFFIRSTQ* and *VFNUMQ* fields in the VPLAN_RX_QBASE register. VFs that are allocated contiguous receive queues in the PF space can get up to 256 receive queues.
- Non-statically allocated VFs and VMDq2 VSIs are allocated up to 16 "scattered" receive queues in the PF space by the VPLAN_RX_QTABLE, and VSILAN_QTABLE registers. The VSILAN_QTABLE registers enable scattered receive queue allocation useful for dynamic VM motion.
  - For VFs, the VSILAN_QTABLE of all its VSIs must be set to the same indexes assigned to the VF by the VPLAN_RX_QTABLE.
  - These tables are not readable for the VFs. Instead, the VF is notified by the PF about the number of its allocated receive queues (using a software API or the mailbox). The PF on its side, must allocate the receive queues starting at receive queue zero and setting contiguous entries in these tables.
- VSIs allocated to nominal PF traffic or PF control ports are expected to be statically allocated and therefore could be contiguous. Contiguous space is defined by the *VSIBASE* field in the VSILAN_QBASE register. VSIs that are allocated contiguous receive queues in the PF space can get any number of receive queues.
  - The size of the contiguous VSI is not enforced by hardware setting; it is the responsibility of software to keep within the VSI boundaries.
  - During reception, hardware checks that the queue index generated by the receive classification filters does not exceed the PF queue range. However, it does not check if the queue index exceeds the VSI range. It is the responsibility of PF software to define the receive classification filters so that the queues do not exceed the VSI space.
- Contiguous versus "scattered" VSI is controlled by the *VSIQTABLE_ENA* flag in the VSILAN_QBASE register.
- It is the responsibility of PF software to define "free" receive queues that are within the space of the PF.

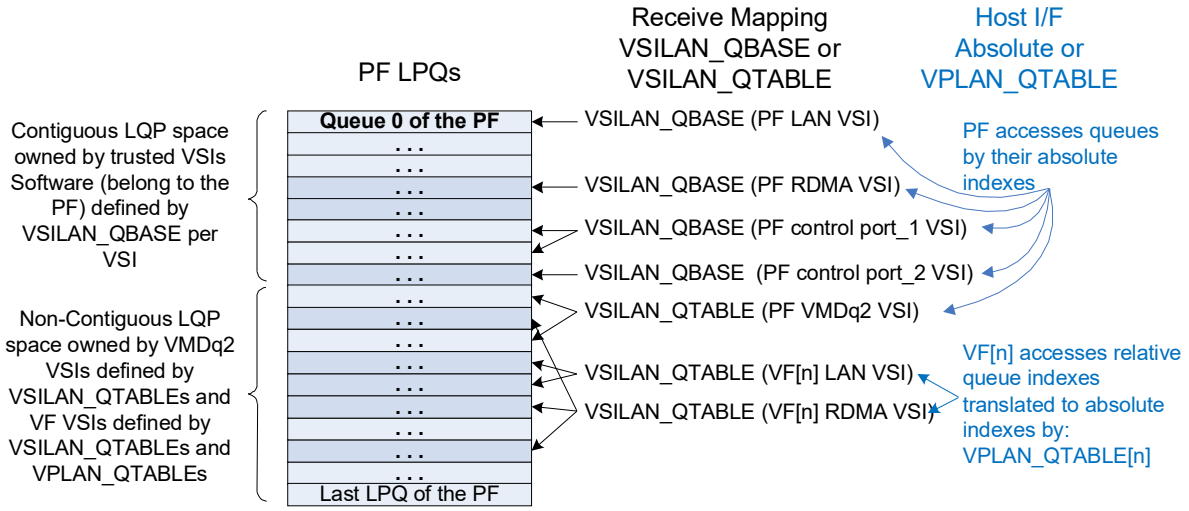


Figure 10-2. PF Queues (Example)

10.2.1.1 Software Access to the Queues of the Functions

PF software accesses its receive queues by their index within the PF space. Queue index "m" of PF "n" equals "m + PFLAN_RX_QALLOC[n].*FIRSTQ*" in the device space. Accessing receive queues outside the PF space does not impact device functionality. Write accesses are terminated successfully on the PCIe bus with no impact on the hardware, while read accesses provide meaningless data.

VF software accesses its receive queues using an index within the VF space. Receive queue indexes are translated to indexes in the PF space by the VPLAN_RX_QTABLE and VPLAN_RX_QBASE registers. The VPLAN_RX_QTABLE supports up to 16 receive queues. When using a contiguous allocation via the VPLAN_RX_QBASE, up to 256 queues can be accessed. A VF might get less than the maximum number of receive queues. The selection between contiguous and scattered is done through the VPLAN_RX_QBASE.*VFQTABLE_ENA* bit. The mapping for a VF in both modes is set through the VPLAN_RXQ_MAPENA.*RX_ENA* bit. A VF software attempt to access receive queues above the allocated ones does not impact device functionality (the same as described above for the PF).

Note: A function (PF or VF) might have multiple VSIs. Still, software accesses all receive queues by their index in the function space rather than index in the VSI spaces.

10.2.1.2 Associating Received Packets to VSI Queues

The E810 associates received packets to VSIs by the embedded switch/ACL as described in Section 7.8. It then associates packets to queues using the classification filters described in Section 7.10.

- As opposed to the software interface, the classification filters assign a queue index in the VSI space. The queue index "n" is mapped to the PF space as follows:
  - Queue index "n" of a contiguous VSI is mapped to queue index "n + VSILAN_QBASE.*VSIBASE*"
  - Queue index "n" of a "scattered" VSI is mapped as follows:
    - For an even "n" it is mapped to VSILAN_QTABLE[n/2].*QINDEX_0*.
    - For an odd "n" it is mapped to VSILAN_QTABLE[(n-1)/2].*QINDEX_1*.
- Invalid receive queues:
  - An invalid queue for contiguous VSI is identified if "n + VSILAN_QBASE.*VSIBASE*" exceeds the PF queue space.
  - An invalid queue "n" for "scattered" VSI is identified if its *QINDEX* in the VSILAN_QTABLE equals 0x3FF, or if "n" is greater than or equal to 16.
- Packets received to invalid queues are dropped and counted by the GLV_RDPC counter of the VSI.

10.2.1.3 LAN Receive Queue Allocation Example

Allocate the receive queues to the VSIs of the PF by setting the VSILAN_QBASE registers, VSILAN_QTABLE and VPLAN_RX_QTABLE. The table below shows an example of five VSIs, where the PF owns 128 receive queues (as configured by PFLAN_RX_QALLOC.*FIRSTQ* = 128 and PFLAN_RX_QALLOC.*LASTQ* = 255):

Table 10-1. Example Receive Queue Allocation

VSI No.	VSI Usage	Requirement	Setting
None	N/A	1 LQP	Allocating dedicated receive queue for the Flow Director Filter Programming Status Descriptors. Queue index 0 in the PF space is absolute queue index 128.
0	PF control port	1 LQP	VSILAN_QBASE[0].*VSIQTABLE_ENA* = 0 // contiguous range VSILAN_QBASE[0].*VSIBASE* = 1 Queue index 1 in the PF space is absolute queue index 129.
1	PF LAN and its VMDq1 VMs	64 LQPs	VSILAN_QBASE[1].*VSIQTABLE_ENA* = 0 // contiguous range VSILAN_QBASE[1].*VSIBASE* = 2 Queue indexes 2...65 in the PF space are absolute queue index 130...193.
2	PF traffic	16 LQPs	VSILAN_QBASE[2].*VSIQTABLE_ENA* = 0 // contiguous range VSILAN_QBASE[2].*VSIBASE* = 66 Queue indexes 66...81 in the PF space are absolute queue index 194...209.
10	VMDq2	2 LQPs	VSILAN_QBASE[10].*VSIQTABLE_ENA* = 1 // scattered range VSILAN_QTABLE[10,0].*QINDEX_0* and *QINDEX_1* = 100 and 102 *QINDEX_0* and *QINDEX_1* in VSILAN_QTABLE[10,1...7] = 0x7FF Queues 100 and 102 in the PF space are absolute queues 228 and 230.
100	VF[20]	4 LQPs	VSILAN_QBASE[100].*VSIQTABLE_ENA* = 1 // scattered range VSILAN_QTABLE[100,0].*QINDEX_0* and *QINDEX_1* = 90 and 98 VSILAN_QTABLE[100,1].*QINDEX_0* and *QINDEX_1* = 110 and 112 *QINDEX_0* and *QINDEX_1* in VSILAN_QTABLE[100,2...7] = 0x7FF VPLAN_RX_QTABLE[20; 0,1,2,3,4:15].*QINDEX* = 90, 98, 110, 112, 0x7FF Queues 90, 98, 110, and 112 in the PF space are absolute queues 218, 226, 238, and 240.

10.2.1.4 LAN Receive Initialization Flow

This section describes the LAN engine initialization flow executed by the PF software driver. It is assumed that a PF reset (PFR) was initiated by the software prior to this flow:

1. The LAN receive queues are allocated to the PF by NVM setting loaded to the PFLAN_RX_QALLOC registers following a core reset (CORER).
2. The statistic counters are cleared only at Power On Reset (POR). As part of the PF driver initialization, the software should read all the PF and its VF statistic counters. The values of these counters is the baseline for any statistics collected later.
3. OS driver only step: Transit the device to non-PXE mode by initiating the Clear PXE Mode admin command. See the command description in Section 10.4.3.2.
4. Most of the LAN engine logic is cleared by the hardware by core reset signal (CORER). If it is not guaranteed that a CORER was initiated, the software should clear the following control registers (listed in the "LAN Transmit Receive Registers" section) of the PF and its VFs. Mapping of receive queues is as follows:
   a. Clear the QRX_CTRL and QRX_TAIL registers of all the PF and its VFs LAN receive queues.

      Note: Writing the value "0'" to the QRX_CTRL register does not clear it. The *Fast_QDIS* and *CDS* bits are zeroed by writing a value of "1" to them.

   b. Initialize LAN port parameters using the Set Port Parameters admin command (setting "save bad packet", default VSI and more). In a single VSI per port, the VSI parameters are loaded from the NVM, while software could execute this step only if it requires other setting then the NVM image.

5. Allocate the receive queues to VFs and VSIs of the PF or each assigned VSI do the following as part of "create VSI" procedure:
   a. Allocate the receive queues to the VSI (see programming example in Section 10.2.1.3).
   b. Program the VSILAN_QBASE and VSILAN_QTABLE (use QBASE option for "contiguous" receive queues or the QTABLE option for "scattered" receive queues).
   c. Program the VPLAN_RX_QTABLE and VPLAN_RX_QBASE of each assigned VF.
6. Enable individual receive and transmit queues of the PF and its VFs following the flow described in Section 10.4.3.1.1 and Section 10.5.5.1, respectively.

10.2.2 LAN Transmit Queue Allocation

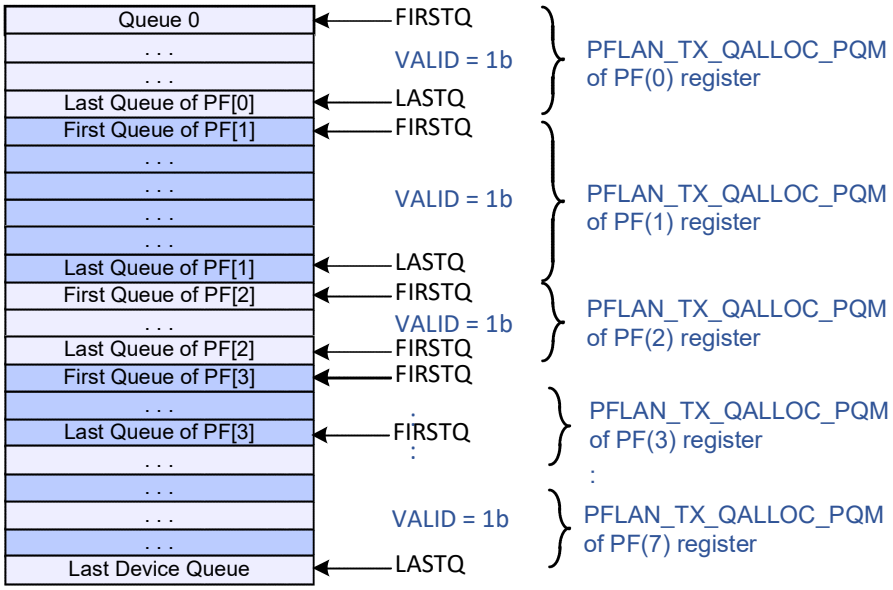


Figure 10-3. LAN Transmit Queues Allocation Example of 8 PFs

The PF software device driver is expected to read the values of the PFLAN_TX_QALLOC register (auto-loaded from NVM at reset) to determine the transmit queues owned. The PF allocates these transmit queues by programming the VPLAN_TX_QTABLE and VPLAN_TX_QBASE registers. Note the following:

- VFs used by the VMM are expected to be statically allocated and therefore could be contiguous. Contiguous space is defined by the *VFFIRSTQ* and *VFNUMQ* fields in the VPLAN_TX_QBASE register. VFs that are allocated contiguous transmit queues in the PF space can get up to 256 transmit queues. The selection between contiguous and scattered is done through the VPLAN_TX_QBASE.*VFQTABLE_ENA* bit. The mapping for a VF in both modes is set through the VPLAN_TXQ_MAPENA.*TX_ENA* bit.

- Non-statically allocated VFs are allocated up to 16 "scattered" transmit queues in the PF space by the VPLAN_TX_QTABLE registers.
  - These tables are not readable for the VFs. Instead, the VF is notified by the PF about the number of its allocated transmit queues (using a software API or the Mailbox). The PF on its side, must allocate the transmit queues starting at transmit queue zero and setting contiguous entries in these tables.
- It is the responsibility of PF software to define "free" transmit queues that are within the space of the PF.

10.2.2.1 Software Access to the Queues of the Functions

PF software accesses its transmit queues by their index within the PF space. Queue index "m" of PF "n" equals "m + PFLAN_TX_QALLOC[n].*FIRSTQ*" in the device space. Accessing transmit queues outside the PF space does not impact device functionality. Write accesses are terminated successfully on the PCIe bus with no impact on the hardware while read accesses provide meaningless data.

VF software accesses its transmit queues using an index within the VF space. Transmit queue indexes are translated to indexes in the PF space by the VPLAN_TX_QTABLE and VPLAN_TX_QBASE registers. The VPLAN_TX_QTABLE supports up to 16 transmit queues. When using a contiguous allocation via the VPLAN_TX_QBASE, up to 256 queues can be accessed. A VF might get less than the maximum number of transmit queues. A VF software attempt to access transmit queues above the allocated ones does not impact device functionality (the same as described above for the PF).

Note: A physical function might have multiple VSIs. Still, software accesses all transmit queues by their index in the function space rather than index in the VSI spaces.

10.2.3 Dynamic Queue Allocation in Rx and Tx

The E810 supports dynamic queue allocation between VSIs of each PF. Dynamic queue allocation is mainly aimed to support dynamic load balance of VFs according to needs due to VM motion or any other queuing load balancing scheme. Queues can be deallocated from an active VSI during run time. These queues can then be allocated to another VSI of the same VF or to another VF. The whole flow is managed by the PF as detailed below.

Both receive and transmit queues can be dynamically and independently allocated.

The following steps relate to receive queues of VFs while it is similar to receive queues of VMs:

1. PF software communicates to the VF that it needs to give up a specific receive or transmit queue(s).
2. The PF removes the queue(s) from the VF/VM by updating VPLAN_RX_QTABLE(VF), VPLAN_RX_QBASE(VF).*VFNUMQ* for Rx VF queues, calling Update VSI (Section 7.8.12.3.2) for Rx VM queues, or by updating VPLAN_TX_QTABLE and VPLAN_TX_QBASE(VF).*VFNUMQ* for the VF transmit queue case. As a result, the VF/VM can no longer access the queue(s).

   Note: The fields *VFNUMQ* and *VFQTABLE_ENA* of registers VPLAN_RX_QBASE(VF) and VPLAN_TX_QBASE(VF) must not be changed dynamically. Their setting is done only as part of the VF establishment flow.
3. PF software disables the relevant queues, following "queue disable" flow described in Section 10.4.3.1.2 for receive queues, or in Section 10.5.5.8.3 for transmit queues. As part of the flow, the PF waits for the hardware indication that the queues are disabled with no further activity to host memory.
4. The PF notifies the VF that it can release the host memory structures of the removed queue(s).

5. The PF remaps these queue(s) to another VF as follows:
   a. It programs the new queue and enables these queues (see Section 10.4.3.1.1 for receive queue enablement flows, and Section 10.5.5.8.1 for transmit queue enablement flows).
   b. It maps the queue(s) to the new VF or VM by updating VPLAN_RX_QTABLE(VF), VPLAN_RX_QBASE(VF).*VFNUMQ* for Rx VF queues, calling Update VSI (Section 7.8.12.3.2) for Rx VM queues, or by updating VPLAN_TX_QTABLE and VPLAN_TX_QBASE(VF).*VFNUMQ* for the VF transmit queue case.

      Note: The fields *VFNUMQ* and *VFQTABLE_ENA* of registers VPLAN_RX_QBASE(VF) and VPLAN_TX_QBASE(VF) must not be changed dynamically. Their setting is done only as part of the VF establishment flow.

   c. The PF informs the VF or VM about this action.
6. The queues are ready to be used by the VF.

10.2.4 LAN Transmit Completion Queue and Doorbell Queue Allocation

Completion Queues and Doorbell sharing among PF mechanism is similar to Transmit and Receive Queue allocation mechanism. The PF software device driver is expected to read the values of the PFLAN_CP_QALLOC and PFLAN_DB_QALLOC registers (auto-loaded from NVM at reset) to determine the Completion and Doorbell Queues owned.

The PF can allocate those Completion Queues and Doorbell Queues or part of them for VFs usage. Allocating of a Completion Queue to a VF does not require any special Hardware CSR setting, since Completion Queue run time operation does not include doorbell.

Doorbell Queues allocation to VFs is programmed in CSR VPLAN_DB_QTABLE. Up to four Doorbell Queues can be associated with a VF.

10.2.4.1 Software Access to the Queues of the Functions

PF software accesses its Completion and Doorbell Queues by their index within the PF space. Completion Queue index "m" of PF "n" equals "m + PFLAN_CP_QALLOC[n].*FIRSTQ*' in the device space. Accessing Completion or Doorbell Queues outside the PF space does not impact device functionality. Write accesses are terminated successfully on the PCIe bus with no impact on the hardware, while read accesses provide meaningless data.

VF software accesses its Doorbell Queues using an index within the VF space. Doorbell Queue indexes are translated to indexes in the PF space by the VPLAN_DB_QTABLE registers. The VPLAN_DB_QTABLE supports up to four Doorbell Queues. A VF software attempt to access Doorbell Queues above the allocated ones does not impact device functionality (the same as described above for the PF).

Note: A function might have multiple VSIs. Still, software accesses all transmit queues by their index in the function space rather than index in the VSI spaces.

10.3 Steering Tag and Processing Hint Support for LAN Engine Traffic (TPH)

See Section 3.1.2.6.2 for information on TLP processing hint support.

Table 10-2 describes how steering tag and processing hints are generated and how TPH operation is enabled for types of DMA traffic associated with the LAN queues.

Table 10-2. Steering Tag and Processing Hint Programming by the LAN Engine

Traffic Access	Steering Tag Value and TPH Enablement	PH Value
Read Receive Descriptor	CPUID and TPHRDesc in the Rx-Queue Context	Desc_PH in GLTPH_CTRL register
Write-Back Receive Descriptor	CPUID and TPHWDesc in the Rx-Queue Context	Desc_PH in GLTPH_CTRL register
Write Receive Packet Payload	CPUID and TPHData in the Rx-Queue Context	DATA_PH in GLTPH_CTRL register
Write Receive Packet Header	CPUID and TPHHead in the Rx-Queue Context	DATA_PH in GLTPH_CTRL register
Read Transmit Descriptor and Quanta Descriptor	CPUID and TPHRDesc in the Tx-Queue Context	Desc_PH in GLTPH_CTRL register
Write-Back Transmit Descriptor	CPUID and TPHWDesc in the Tx-Queue Context	Desc_PH in GLTPH_CTRL register
Transmit Head Write-Back	CPUID and TPHWDesc in the Tx-Queue Context	Desc_PH in GLTPH_CTRL register
Read Transmit Packet	CPUID and TPHRPacket in the Tx-Queue Context	DATA_PH in GLTPH_CTRL register
Read Doorbell Queue Descriptor	CPUID and TPHRDesc in the Doorbell Queue Context	Desc_PH in GLTPH_CTRL register
Write-Back Doorbell Queue Descriptor	CPUID and TPHWDesc in the Doorbell Queue Context	Desc_PH in GLTPH_CTRL register
Write to Completion Queue Descriptor	CPUID and TPHWDesc in the Completion Queue Context	Desc_PH in GLTPH_CTRL register

10.4 LAN Receive Data-Path

The LAN Receive Data-Path sections includes the following major topics:

- Receive packets stored in system memory.
- Indicating free descriptors to the hardware and indicating completed descriptors back to the software.
- Receive descriptor queues which are called also “descriptor ring”.
- Receive arbitration (covered in Section 8.2.1.2.3).
- Stateless receive offloads.

10.4.1 Receive Packet in System Memory

Receive packets are posted to system (host) memory buffers indicated to the hardware by descriptors. There are several types of descriptors detailed in Section 10.4.2; these include pointers to the data buffers and status indications of the received packets. Figure 10-4 shows two examples of receive packets in host memory composed of two buffers (indicated by two matched descriptors). The E810 fetches the receive descriptors (on demand) to an internal cache.

A few rules relating receive packet posting to host memory are:

- Receive packets can span one to five buffers (descriptors).
- Receive packets shorter than 64 bytes are never posted to host memory (even in save bad frame mode - enabled by the *SBP* flag in the PRT_SBPVSI register).

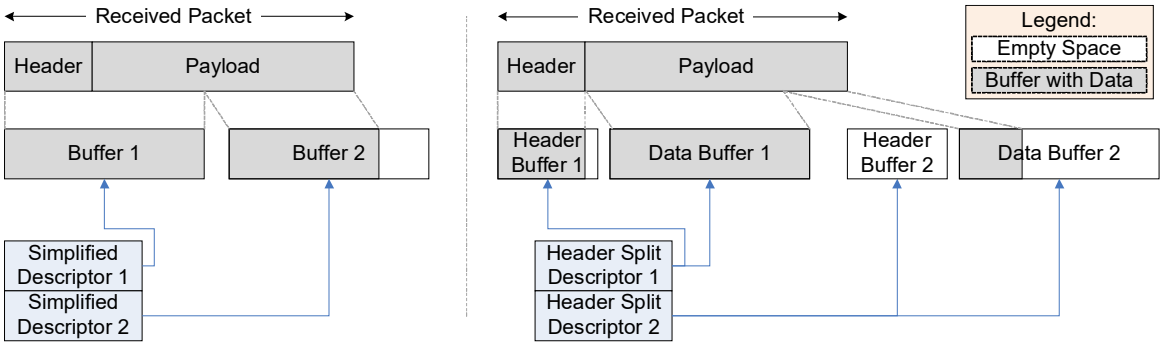


Figure 10-4. Receive Packet in System Memory

10.4.1.1 Receive Descriptor Cache

10.4.1.1.1 Descriptor Fetch Policy

The E810 fetches multiple receive descriptors at a time to minimize PCIe and memory bandwidth overhead; 8 or 4 descriptors when using 16-byte or 32-byte descriptors, respectively. New descriptors are fetched to the cache when there are fewer descriptors than incoming packets require, or the last free descriptor is used for a received packet.

Note the following rules relating descriptor fetch policy:

- Following a CORE Reset, the hardware wakes up in "PXE" mode (the *PXE_MODE* flag in the GLLAN_RCTL_0 register is set (1b)). PXE mode functionality and limitations are:
  - Receive queue length restrictions are detailed in Table 10-12, "LAN Rx-Queue Context (256 Bits = 32 Bytes)" in the *QLEN* field.
  - Software can bump the tail at descriptor granularity. Hardware fetches and writes back these descriptor at descriptor granularity as well.
  - Each packet can span only on a single buffer (in a single descriptor). A receive packet that is larger than a single buffer is reported as "OVERSIZE" in the receive descriptor.
- During nominal performance operation mode, the *PXE_MODE* flag must be cleared by the software by calling the Clear PXE Mode AQ command (0x110). This step is expected to occur during the PF software initialization procedure. In the case of multiple active PFs, only the first PF affects the device setting, while the others do nothing.
- When the *PXE_MODE* flag is cleared (0b), software should bump the tail at whole 8 x descriptors granularity.

10.4.2 LAN Receive Descriptors

10.4.2.1 Receive Descriptor - Read Format

10.4.2.1.1 16-Byte Receive Descriptors Read Format

Described below is the 16-byte receive descriptor read format prepared by the software.

Table 10-3. 16-Byte Receive Descriptors Read Format

Quad Word	63 ... 0
0	Packet Buffer Address
1	Header Buffer Address
	63 ... 0

Packet Buffer Address (Quad Word 0, Bits 0:63, 64 bits)

The physical address of the packet buffer defined in byte units. The packet buffer size is defined by the DBUFF parameter in the receive queue context.

Header Buffer Address (Quad Word 1, Bits 0:63, 64 bits)

The physical address of the header buffer defined in byte units. The header address should be set by the software to an even number (word-aligned address). The *Header Buffer Address* is meaningful only for Header Split queues and Split Always queues as defined by the DTYPE field in the receive queue context. If a received packet spans across multiple buffers, only the first descriptor's header buffer is used. The header buffer size is defined by the HBUFF parameter in the receive queue context.

Note: The LS bit should be set to zero regardless of header split enablement, since it is used for Descriptor Done (DD) indication to the software (as described in the descriptor write-back format).

10.4.2.1.2 32-Byte Receive Descriptors Read Format

Described below is the 32-byte receive descriptor read format prepared by the software.

Table 10-4. 32-Byte Receive Descriptors Read Format

Quad Word	63 ... 0
0	Packet Buffer Address
1	Header Buffer Address
2	Reserved (0x0)
3	Reserved (0x0)
	63 ... 0

The fields in first 16 bytes are Identical to the 16 Byte descriptors described in Section 10.4.2.1.1.

10.4.2.2 Receive Descriptor - Write-Back Format

The following subsections describe the fields of Receive Descriptor write-back when using 16-byte and 32-byte descriptors. In both cases, a single packet might span on a single buffer or multiple buffers reported by their matched descriptors. If a packet is described by a single descriptor, all the fields are valid.

Following are some rules that apply for a packet that is described by multiple descriptors:

- The following fields are valid in all descriptors of a packet:
  - DD flag (Done)
  - EOP flag (End of Packet)
  - PKTL field (Packet content length)
- The following fields are valid only in the first descriptor of a packet:
  - HDRL (Packet content length in the header buffer)
  - SPH (Header is identified for header split functionality)
  - HBO (Header Buffer Overflow)
- All other fields are valid only in the last descriptor of a packet.

10.4.2.2.1 16-Byte Legacy Receive Descriptors Write-Back Format

Described below is the 16-byte receive descriptor write-back (WB) format.

Table 10-5. 16-Byte Receive Descriptors Write-Back Format

Quad Word	63 … 32	31 … 16	15	14	13 … 8	7 … 0
0	Filter Status	L2TAG1	RSV	MIRR	RXDID	
1	Length	PTYPE	RSV	Error	Status	

Quad Word	63 … 38	37 … 30	29	27	26 … 19	18 … 0
0						
1						

RSV (Quad Word 0, Bits 14:15, 2 bits; Quad Word 1, Bits 27:29, 3 bits)

Reserved.

Status (Quad Word 1, Bits 0:18, 19 bits)

Bit(s)	Name	Description
0	DD	Descriptor done indication flag.
1	EOP	End of packet flag is set to 1b indicating that this descriptor is the last one of a packet.
2	L2TAG1P	L2 TAG 1 presence indication while the L2 Tag is stripped from the packet and reported in the L2TAG1 field in the descriptor. The type of the Tag is defined by the *L2TSEL* flag in the queue context. The *L2TSEL* flag selects between the first or second active flags in the *SHOWTAG* field in the VSI_TSR register of the VSI. The structure of this tag is defined by the matched GL_SWT_L2TAG*xxx* registers. If the specified L2 tag is not present in the packet the *L2TAG1P* flag is cleared.
3	L3L4P	For IP packets, this flag indicates that detectable L3 and L4 integrity check is processed by the hardware. See Section 10.4.4.3 for details on which headers are processed and how.

Bit(s)	Name	Description
4	CRCP	CRCP indicates that the Ethernet CRC is posted with data to the host buffer. ***Note:*** Strip CRC is enabled by the *CRCStrip* flag in the queue context. If the *RXE* error flag is set, the CRC bytes are not stripped regardless of the *CRCStrip* flag in the queue context. Loopback packets originated by another local VSI for which the hardware computes the CRC are never posted with the CRC bytes regardless of the CRCStrip setting in the queue context.
5:7	Reserved	Reserved. Set to "0" by hardware.
8	EXT_UDP_0	This flag is set for received tunneled packets with outer UDP header on which the outer UDP checksum word equals to zero. ***Note:*** UDP checksum zero is an indication that there is no checksum. This option is valid only for IPv4 packets and considered an exception error for IPv6 packets (reported to the stack by the driver).
9:10	UMBCAST	Destination Address can be one of the following: 00b = Unicast 01b = Multicast 10b = Broadcast 11b = Mirrored Packet
11	FLM	Flow Director filter match indication. This flag is set if the received packet matches any of the Flow Director (FD) filters that direct the packet to a specific receive queue. See also the description of the *FLTSTAT* field below.
12:13	FLTSTAT	The *FLTSTAT* indicates the reported content in the "Filter Status" field (see conditions in the "Filter Status" field): 00b = No Data in the filter status field (The packet does not meet any of the cases below). 01b = FD filter ID (this option is valid only for 16B descriptor while in 32B it is reported elsewhere). 10b = Reserved. 11b = Hash filter signature (RSS).
14	LPBK	Loopback indication, which means that the packet is originated from this system rather than the network.
15	IPV6EXADD	Set when an IPv6 packet contains a Destination Options Header or a Routing Header. If the packet contains two IPv6 headers (tunneling), the *IPV6EXADD* is a logic 'OR' function of the two IP headers.
16:7	Reserved	Reserved.
18	INT_UDP_0	This flag is set for received UDP packets on which the UDP checksum word equals zero. ***Note:*** UDP checksum zero is an indication that there is no checksum. This option is valid only for IPv4 packets and considered an exception error for IPv6 packets (reported to the stack by the driver). ***Note:*** For tunneled packets with UDP header, this flag relates to the checksum field in the inner UDP header.

Error (Quad Word 1, Bits 19:26, 8 bits)

Bit(s)	Name	Description
0	RXE	The *RXE* error bit is an indication for any of the following MAC errors: • CRC error. • Alignment error. • Oversize error. • Undersizes error. • Length error. Packets with *RXE* are posted to host memory to Rx-Queue 0 of the VSI defined by the PRT_SBPVSI register if enabled by the *SBP* flag in the same register. If the *RXE* flag is set then any other status fields reporting the content of the packet are meaningless.
1	Reserved	Reserved.

Bit(s)	Name	Description
2	HBO	Header Buffer Overflow. This flag is set when using Header split buffers or Split Always buffers and the identified packet header is larger than the header buffer.
3:5	L3L4E / FCE	For IP packets processed by the hardware, the *L3L4E* flag has the following encoding: Bit 3 = IPE: IP checksum error indication (for tunneled packets it is the most inner IP header indication). Bit 4 = L4E: L4 integrity error indication (most inner L4 header in case of UDP tunneling). Bit 5 = EIPE: External (most outer) IP header or UDP checksum error. ***Note:*** For the purpose of these bits, a tunneled packet is a VXLAN/GRE/Geneve packet.
6	OVERSIZE	Oversize packet error indicates that the packet is larger than 5 descriptors in nominal operation or larger than 1 descriptor in PXE mode. In this case, the portions of the packet that exceeds the permitted number of descriptor(s) is not posted to host memory.
7	Reserved	Reserved.

RXDID (Quad Word 0, Bits 0:7, 8 bits)

Rx Write-Back Descriptor Type ID:

0x00 - 0x01 = Legacy Descriptor
0x02 - 0x06 = Flexible Descriptor
0x07 - 0x07 = Mirrored Packet
0x08 - 0x3F = Flexible Descriptor
0X40 - 0x7F = Reserved

MIRR (Quad Word 0, Bits 8:13, 6 bits)

Matched Mirror Rule ID that directed the packet to this queue. If the packet is not mirrored, this field equals zero.

L2TAG1 (Quad Word 0, Bits 16:31, 16 bits)

Stripped L2 Tag from the receive packet. This field is valid if the *L2TAG1P* flag in this descriptor is set (see additional description of the *L2TAG1P* flag in the *Status* field).

Filter Status (Quad Word 0, Bits 32:63, 32 bits)

Multiplexed field between RSS (hash filter), FD Filter ID as indicated by the *FLTSTAT* field. Note that the FD filter ID is reported in this field only in 16-byte descriptors. In 32-byte descriptors, the data is reported in a different field.

- If the packet matches a FD filter that enables its FD filter ID reporting while using a 16-byte descriptor, *FLTSTAT* equals 01b and this field contains the programmed FD filter ID.
- Else, if the packet matches the Hash filter, *FLTSTAT* equals 11b and this field contains the hash signature (RSS).
- Else, *FLTSTAT* equals 00b and this field is set to zero.

Length (Quad Word 1, Bits 38:63, 26 bits)

Bit(s)	Name	Description
0:13	PKTL	Packet content length in the packet buffer defined in byte units.
14:24	HDRL	Packet content length in the header buffer defined in byte units.
25	SPH	The Split Header flag is an indication that the device identified the packet header. See Section 10.4.4.2 for a complete description of packet types identified for header split and conditions for usage of the header and data buffers.

PTYPE (Quad Word 1, Bits 30:37, 8 bits)

Packet Type field encode supported packet types as listed in Table A-1.

Note: For Packet Types 256 and above, Flexible descriptors must be used. See Section 7.6.

10.4.2.2.1.1 Dummy Receive Descriptors

In some cases, the hardware might reserve one more descriptor than needed for a given packet. In this case, this dummy descriptor is written back as the last descriptor of the packet with a data length field == 0. For this packet, the dummy descriptor carries all the flags that are normally written at the last descriptor of a multi-descriptors packets.

10.4.2.2.2 32-Byte Receive Descriptors Write-Back Format

The 32-byte descriptor is composed of four quad words. The first two quad words are identical to the 16-byte descriptor write-back other than the Flow Director Filter ID that is moved from the "Filter Status" field to the "FD Filter ID / Flexible Bytes High" field.

Section 10.4.2.2.1.1 describes to concept of Dummy Descriptor in 16-byte descriptor format. Dummy Descriptor is relevant to 32-Byte descriptor write-back as well.

Table 10-6. 32-Byte Receive Descriptors Write-Back Format

Quad Word	63:48	47:38	37:32	31:30	29:26	25:19	18:16	15	14:13	12:8	7:0
0	Filter Status			L2TAG1				RSV	MIRR		RXDID
1	Length			PTYPE		RSV	Error		Status		
2	L2TAG2 (2nd)			L2TAG2 (1st)			Reserved				Ext_Status
3	FD Filter ID						Reserved				

	63:48	47:32	31:18	17:12	11:0

Ext_Status (Quad Word 2, Bits 0:11, 12 bits)

Bit(s)	Name	Description
0	L2TAG2P	L2 TAG 2 presence indication while the L2 Tag is stripped from the packet to the *L2TAG2 (1st)* field in the descriptor. The type of the Tag is defined by the *L2TSEL* bit in the queue context. This flag selects between the first or second active flags in the *SHOWTAG* field in the VSI_TSR register of the VSI. The structure of this tag is defined by the matched GL_SWT_L2TAG*xxx* registers. If the stripped tag is larger than one word, the first word is posted to the *L2TAG2 (1st)* field and the second word is posted to the *L2TAG2 (2nd)* field. If it is a single word, it is stored in the *L2TAG2 (2nd)* field. If the specified L2 tag is not present in the packet the *L2TAG2P* flag is cleared.
1:3	Reserved	Reserved.
4:5	FLEXBH_STAT	The *FLEXBH_STAT* field indicates the content of the *FD Filter ID/Flexible Bytes High* field as follows: 00b = The field is set to zero. 01b = FD filter ID is reported in the *FD Filter ID* field. 10b = Reserved. 11b = Reserved.
6:8	Reserved	Reserved.

Bit(s)	Name	Description
9	FDLONGB	The *FDLONGB* flag is set if the matched FD filter's index within its bucket is above threshold. If the packet is searched in the FD filter and it is not found, the *FDLONGB* represents the bucket length relative to the same threshold. The threshold is defined by the *FDLONG* parameter in the GLQF_FD_CTL register.
10:11	Reserved	Reserved.

L2 Tags (Quad Word 2, Bits 32:63, 32 bits)

Bit(s)	Name	Description
0:15	L2TAG2 (1st)	Extracted L2 Tag 2 from the packet (see *L2TAG2P* flag in the *Ext_Status* field).
16:31	L2TAG2 (2nd)	Extracted second word of the L2 Tag 2 from the packet (see *L2TAG2P* flag in the *Ext_Status* field).

FD Filter ID (Quad Word 3, Bits 32:63, 32 bits)

If the packet matches a FD filter that enables reporting the *FD filter ID*, then *FLEXBH_STAT* equals 01b and this field contains the programmed FD filter ID. Otherwise, *FLEXBH_STAT* equals 00b and this field is set to zero.

10.4.2.2.3 Programming Status Descriptor Write-Back Format

The programming status descriptor provides an indication of FD filter programming.

Quad Word 0

Bit(s)	Name	Field Size	Description
0:7	RXDID	8	Descriptor Type. For Filter Programming WB it equals 0x40
8:27	Reserved	1	Reserved. Set to 0x0.
28:31	Bucket_LEN	4	The filter location within the bucket capped at 15.[1]
32:63	Filter_Status	32	The *Filter_Status* reports back the programmed FD filter ID.

1. The Bucket_LEN parameter is reported only if enabled by CSR GLQF_FD_CTL. In addition, the Bucket_LEN parameter is reported only for programming or removal requests that are completed successfully.

Quad Word 1

Bit(s)	Name	Field Size	Description
0	DD	1	Descriptor Done flag.
1:2	PROG_ID	2	The *PROG_ID* indicates the status reported by this descriptor as follows: 00b = FD filter add status. 01b = FD filter remove status. 10b = Reserved. 11b = Reserved.
3	Reserved	1	Reserved.
4	FAIL	1	FD filter programming failed due to no space in the table or the function exhausted its quota. Or, FD filter removal failed that could be a result of an attempt to remove non-existent filter entry.
5	FAIL_PROFILE		FD filter programming failed due to no matched profile or profile with an empty Field Vector.

Bit(s)	Name	Field Size	Description
6:7	Reserved	2	Reserved.
8:21	FLT_Addr	14	Filter entry address in the FD table (measured in filter entry units).[1]
22:27	Reserved	6	Reserved.
28:34	PKT_PROF	7	The packet profile that was identified by the device at filter programming.
35:37	Reserved	3	Reserved.
38:63	Bucket_HASH	26	26 LS bits of the hash function on the input set that is used for bucket index.[1]

1. The Bucket_HASH and the FLT_Addr parameters are reported only if enabled by CSR GLQF_FD_CTL. In addition, the FLT_Addr parameter is reported only for programming or removal requests that are completed successfully.

Note: If the queue is configured for 32-byte descriptors, this descriptor is padded by all zero two quad words.

10.4.2.2.4 Receive Flexible Advanced Descriptors Format

The flexible descriptor format is and its programming are detailed in Section 7.6.

10.4.3 LAN Receive Queue (Ring)

Received packets are posted to host memory through a set of queues. Each queue is a cyclic ring made of a sequence of receive descriptors in contiguous memory. These queues are also called "descriptor rings". The E810 supports up to 2048 receive queues allocated to PFs and VFs.

Receive queues are defined by a set of parameters called the "queue context". The main parameters are the queue pointers presented in Figure 10-5. The queue context includes additional parameters that define the queue functionality, as detailed in Section 10.4.3.6. All context parameters are kept in device registers accessible to the PF, while only the Tail registers (needed at run time) are accessible to the VFs.

The software interface to the queue for its initialization, during nominal operation as well as queue disable flow, is described in Section 10.4.3.1. The E810 includes additional global setting option parameters for the whole device or per function that affect multiple queues described in Section 10.4.3.1.

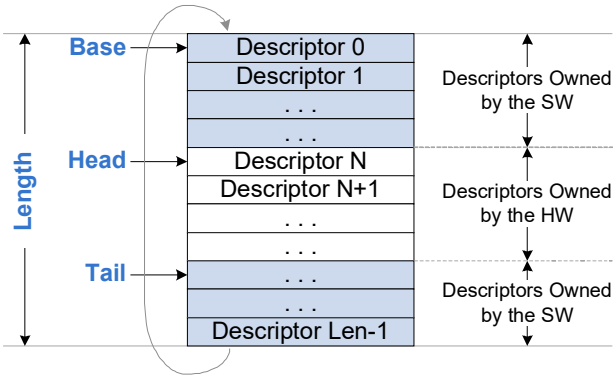


Figure 10-5. Receive Descriptor Ring Structure

10.4.3.1 Receive Queue Programming

Queue enable and disable flows are described in the following subsections.

10.4.3.1.1 Receive Queue Enable Flow

Table 10-7. Receive Queue Enable/Disable Flags

QRX_CTRL[n] Register QENA_* Bits		Receive Queue State
QENA_REQ	QENA_STAT	
0	0	Queue is not enabled.
1	0	Queue Enable request by the software.
1	1	Queue is enabled.
0	1	Queue Disable request by the software.

Queue enable flow is executed by the PF software for PF queues as well as for the queues of its VFs. The flow below assumes that the queue is already allocated as detailed in Section 10.2.

Software steps:

1. The function that owns the queue (PF or VF) allocates contiguous memory in its own memory space for the receive ring. It then program the receive descriptors in the ring so they are ready for new packet reception.
2. Program the Rx-Queue context parameters in both RLAN and RCB structures. Receive queue context structures are detailed in Section 10.4.3.6.
3. Clear the Tail pointer in the QRX_TAIL[n] register (in RLAN Queue Context) and then set the Tail pointer to the end of the descriptor ring, where "n" is the queue index within the PF space.
4. Set the *QENA_REQ* flag in the QRX_CTRL[n] register, where "n" is the queue index within the PF space (in RCB Queue Context).
5. If the Rx-Queue is associated with a No-Drop TC, then:
   a. Set *CDS* bit as well. This resets its state to init state.
   b. Set or reset the *CDE* bit according the queue's fast response policy.
      - QRX_CTRL register structure bits *CDE* and *CDS* are used for a QoS configuration of the queue.
6. As a response, the hardware sets the *QENA_STAT* flag in the QRX_CTRL[n] register. The *QENA_STAT* follows the *QENA_REQ* almost instantly and not more than 10 μs after that. Once the *QENA_STAT* flag in the QRX_CTRL[n] register is set, software can start using the queue.
7. If the queue is targeted for a VF, PF software should also program the matched entry in the VPLAN_RX_QTABLE. Then it is expected to inform the VF software that the queue is enabled.

10.4.3.1.2 Receive Queue Disable Flow

Queue disable flow is executed by PF software for PF queues as well as for the queues of its VFs.

1. Remove any filtering or switching rule which directs packets to this queue (e.g. RSS, Flow Director, and so on).
2. Wait for configuration completion.
3. Remove the queue from the interrupt linked list as described in Section 9.1.3.1.2.
4. The PF software clears the *QENA_REQ* flag in the QRX_CTRL[n] register, where "n" is the queue index within the PF space.
5. The hardware generates a "queue disable" marker to the receive "pipe".
6. Eventually, the "queue disable" marker gets to the top of the "pipe". At this point, it is guaranteed that the "pipe" does not contain any additional receive packets for the specific queue.
7. The hardware waits for completion of all outstanding requests from the specific queue on the PCIe bus.
8. The RCB queue context *QENA_STAT* flag in the QRX_CTRL[n] register is cleared.
9. Once the *QENA_STAT* flag in the QRX_CTRL[n] register is cleared, software can release all memory structures of the queue.

10.4.3.1.2.1 Fast Receive Queue Disable Flow

Fast queue disable flow should be executed by software only as part of a VF reset flow (or VM reset flow). Following a PFR, the device does all this automatically.

1. It is assumed that a VFR was initiated by the PF software and the matched *VFRD* flag in the register is already active. Or, a VMR was initiated by PF software and the matched VMRD flag in the VSIGEN_RSTAT register is already active.
2. The PF software sets the *FAST_QDIS* flag (and clear the *QENA_REQ* flag) in the QRX_CTRL[n] register, where "n" is the queue indexes in the PF space for all VF or VM queues.
3. The RCB queue context *QENA_STAT* flag in the matched QRX_CTRL[n] register is cleared.

10.4.3.2 Clear PXE Mode Admin Command (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 10-8. Clear PXE Mode Admin Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0110	Command opcode.
Datalen	4-5	0x00	N/A (reserved zero).
Return Value/VFID	6-7	0x00	N/A (reserved zero).
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16	0x2	Reserved. Set to 0x2.
Reserved	17-31		Reserved.

Table 10-9. Clear PXE Mode Admin Command Completion

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0110	Command opcode.
Datalen	4-5	0x00	N/A
Return Value/VFID	6-7		Some comments on specific errors (see Section 9.5.10 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 the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-31		Reserved.

10.4.3.3 Configure No-Drop Policy Admin Command (0x00112)

The Configure No Drop Policy admin command configures the device to use the "No Drop" policy, and it might be used at PXE mode. The structure of the admin command and its completion are as follows:

Table 10-10. Configure No-Drop Policy Admin Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0112	Command opcode.
Datalen	4-5	0x00	Must be zero. Value is ignored.
Return Value/VFID	6-7	0x00	Must be zero. Value is ignored.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Force No-Drop	16	0x2	Sets the No-Drop/Drop policy: 0b = Remove the force and return to normal firmware behavior. 1b = Force No-Drop. ***Note:*** At the admin command response for this bit is defined as reserved, making all of Byte 16 reserved at the response buffer.
Reserved	16.1-31		Reserved.

Table 10-11. Configure No-Drop Policy Completion Buffer

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 9.5.5.2.2 for details.
Opcode	2-3	0x0112	Command opcode.
Datalen	4-5	0x00	N/A
Return Value/VFID	6-7		Return value. Zeroed by device driver. Written by firmware. 0x0 = Command success EPERM is returned if the command was called while not in PXE mode. EPERM is returned if software control of the drop/no-drop policy is locked by NVM setting.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Reserved	16-31		Reserved.

Command Operation:

1. The Configure No-Drop Policy AQ command is executed only when the hardware is in PXE mode (GLLAN_RCTL_0.*PXE_MODE* is 0x1), and when the PXE Mode *No-Drop Policy Supported* NVM bit is 1b. Otherwise, the firmware returns EPERM error code.
2. The firmware must maintain an internal variable per port (sw_force_no_drop) that holds the requested No-Drop policy from software. The internal variable is updated as follows:
   a. Initial value, set also after CORER, must be 0.
   b. When the Configure No-Drop Policy AQ command is called, the variable is updated to the received *No-Drop Policy* field.

      Note: The software can disable the No-Drop policy by setting this field to 0.

   c. When a Clear PXE Mode AQ command is called, the variable is set to 0 for all ports.

3. Configure No-Drop Policy AQ command execution:
   a. If Force No-Drop is 1 and sw_force_no_drop is currently 0:
      - Firmware sets the corresponding bits of the port in GLDCB_RTC2PFC_RCB.*TC2PFC* (4 bits or 8 bits according to the link topology). In LFC mode, only the bits that correspond for TC0 should be set.
      - Firmware changes GLDCB_RSPMC.*PFCTIMER* to pxe_pfc_timer.
   b. If Force No-Drop is 0 and sw_force_no_drop is currently 1:
      - Firmware reads the bits of current port from GLDCB_TC2PFC and copies them to the corresponding bits in GLDCB_RTC2PFC_RCB.*TC2PFC*. and then writes the value to GLDCB_RTC2PFC_RCB to align all Rx copies of the CSR.
      - Firmware restores GLDCB_RSPMC.PFCTIMER to the NVM default value.
4. After any update of GLDCB_TC2PFC (the master), if sw_force_no_drop is set for this port, the firmware sets the corresponding bits of the port in GLDCB_RTC2PFC_RCB.*TC2PFC* (4 bits or 8 bits based on link topology).
5. When executing the Clear PXE Mode AQ command, firmware applies (3b) to all ports for which sw_force_no_drop is currently 1.

10.4.3.4 Transitioning Flow to non-PXE Mode

10.4.3.4.1 Device Response to Clear PXE Mode Admin Command

1. When the Clear PXE Mode admin command is initiated, the device checks the value of the *PXE_MODE* flag in the GLLAN_RCTL_0 register.
2. If the *PXE_MODE* flag is found cleared then:
   a. Return a command completion with “EEXIST” indication.
3. Else, the *PXE_MODE* flag is active:
   a. Disable Rx-Queue 0 and Rx-Queue 1 of all enabled PFs by clearing the *QENA_REQ* flag in the QRX_CTRL[0] and QRX_CTRL[1] registers of the PFs.
   b. Wait until all receive queues are disabled by polling the *QENA_STAT* flag in the QRX_CTRL registers of all above Rx-Queues. It is expected that all Rx-Queues are disabled within a few microseconds.
   c. Clear the *PXE_MODE* flag in the GLLAN_RCTL_0 register.
   d. Flow is completed by posting a command completion with “No Error”.

10.4.3.4.2 Software Steps Transitioning to non-PXE Mode

1. The Admin Queue must be active before the following steps.
2. Software initiates the Clear PXE Mode admin command and waits for its completion.
3. Proceeds with the software initialization flow.

10.4.3.5 Receive Queues Doorbells and Completions

10.4.3.5.1 Receive Descriptors and Tail Bump

During nominal operation, the function that owns the queue (PF or VF) accesses the hardware directly.

- Prepare receive descriptors by clearing the *DD* bit and setting the buffer pointer(s). Start at the descriptor indicated by the *TAIL* pointer in the relevant QRX_TAIL register.
- The software should never set the *TAIL* to a value above the descriptors owned by the hardware minus 1. The descriptors considered as "owned by the hardware" are those ones already indicated to the hardware but not yet reported as completed.
- Bump the *TAIL* to the last prepared descriptor plus one.
- The number of "free" descriptors owned by the hardware is defined by *TAIL* minus the *HEAD*. If the number of "free" descriptors becomes lower than *LRXQTRESH*, then an immediate interrupt is triggered. The *HEAD* and *LRXQTRESH* are listed in the Rx-Queue context detailed in Section 10.4.3.6.

10.4.3.5.2 Receive Descriptor Reporting (Descriptor Write-Back)

The E810 reports a completion of receive packet in host memory by status indication in the receive descriptor(s) of the packet - descriptor write-back (WB). The E810 writes back completed descriptor status in one of the following cases:

- Whole line of descriptors (4x32-byte descriptors or 8x16-byte descriptors) are completed.
- All completed descriptors of a queue are evicted from the internal cache.
- Upon assertion of the interrupt associated with the queue.
- If queue is configured for "No Expire" (as explained in Section 10.4.3.6.1), every completed descriptor is written back immediately.

Most applications use interrupts to invoke the driver. Before initiating the interrupt, the E810 posts all completed descriptors of the queue that might be kept in the device caches. Following an interrupt assertion, the device masks any further interrupts preventing interrupt nesting (as explained in Section 9.1.1.3). When the interrupts are re-enabled (by the software), further interrupts that trigger additional write-back of completed descriptors can be initiated.

In some applications, the E810 is activated in polling mode (with no interrupts). Configuring this option is done by following the "WB ON ITR" option in Table 9-1 on page 1155.

10.4.3.6 Receive Queue Context Parameters

This section describes the setting options of the LAN receive queue parameters named as "queue context". The receive queue context is an on-die structure that contains the configuration, state, and internal scratch pad of each one of the receive queues. The receive queue context is used and managed by several blocks in the Rx-Pipe. For easy access of each block, the Rx-Queue context is divided into two sections, the Receive Queue Context section and the Receive Misc. section.

10.4.3.6.1 Receive Queue Context

Receive queue context parameters are initialized by software and then managed by hardware.

The Rx-Queue context consists of two major sections:

- Queue Configuration section.
- Dynamic section.

The first one is programmed by software during the queue init flow. Software does not access this section on runtime. The dynamic section is accessed by software on runtime and therefore it is implemented as dedicated registers vectors.

10.4.3.6.1.1 Receive Queue Context - Static Section

The queue configuration section parameters reside in the QRX_CONTEXT[n, 0...7] registers and are detailed in Table 10-12. The dynamic section is detailed in Section 10.4.3.6.1.2.

The queue context is a set of 256 bits (32 bytes), where bit "i" in the vector is mapped to - bit number {"i" modulo 32} in QRX_CONTEXT register index {Whole number of ("i"/32)} of the queue. For example, the *QLEN* parameter in bits 89 to 101 of the queue context is stored in bit number 25 in QRX_CONTEXT register index 2 to bit number 5 in QRX_CONTEXT register index 3 of the queue.

Table 10-12. LAN Rx-Queue Context (256 Bits = 32 Bytes)

Alias	Width (bits)	LS Bit	MS Bit	Type	SW Init	Description
HEAD	13	0	12	Dynamic	0x0	**Receive Queue Head** An index relative to the beginning of the queue that defines the next descriptor to be used. During idle time, all descriptors starting by the HEAD up to (excluding) the RTAIL are owned by the hardware and the rest are owned by software. During dynamic operation it is not guaranteed that all descriptors below the HEAD are completed.
CPUID	8	13	20	Dynamic	0x0	**CPU Socket ID for TPH** The CPU socket ID is updated by the hardware.
Queue_Block Indication	1	21	21	Dynamic	0b	**Queue Block** When set, the device silently drops packets that arrive to this queue. This bit is set by hardware as a response to a malicious event or during Function Reset flow. This bit must be cleared by the PF before the queue is reused.
Reserved	11	22	31	N/A	0x0	Reserved. Must be set to zeros.
BASE	57	32	88	Static	BASE	**Receive Queue Base Address** Indicates the starting address of the descriptor queue defined in 128-byte units.
QLEN	13	89	101	Static	QLEN	**Receive Queue Length** Defines the size of the descriptor queue in descriptors units from eight descriptors (*QLEN*=0x8) up to 8K descriptors minus 32 (*QLEN*=0x1FE0). QLEN Restrictions: When the *PXE_MODE* flag in the GLLAN_RCTL_0 register is cleared, the *QLEN* must be whole number of 32 descriptors. When the *PXE_MODE* flag is set, the *QLEN* can be one of the following options: Up to 4 PFs, QLEN can be set to: 8, 16, 24 or 32 descriptors. Up to 8 PFs, QLEN can be set to: 8 or 16 descriptors.

Table 10-12. LAN Rx-Queue Context (256 Bits = 32 Bytes) [continued]

Alias	Width (bits)	LS Bit	MS Bit	Type	SW Init	Description
DBUFF	7	102	108	Static	DBUFF	**Receive Packet Data Buffer Size** The Packet Data Buffer Size is defined in 128-byte units. Must be at least 1 KB and up to 16 KB minus 128 bytes.
HBUFF	5	109	113	Static	HBUFF	**Receive Packet Header Buffer Size** The Header Buffer Size is defined in 64-byte units enabling buffer size up to 2 KB minus 64 bytes.
DTYPE	2	114	115	Static	DTYPE	**Descriptor Type** Descriptor Type as defined in Table 10-13.
DSIZE	1	116	116	Static	DSIZE	**Descriptor Size** 0b = 16-byte descriptors. 1b = 32-byte descriptors.
CRCStrip	1	117	117	Static	CRCStrip	**CRC Strip** Strip the Ethernet CRC bytes before the packet is posted to host memory. ***Note:*** The CRC Strip option works properly only if the whole packet is posted to the data buffer(s) in host memory with no other strip option.
Reserved	1	118	118	Static	0b	Reserved. Must be set to 0b.
L2TSEL	1	119	119	Static	0b	**L2 Tag Select** The *L2TSEL* bit defines the reported L2 Tags in the receive descriptor. 0b = The second L2 Tag defined by the *SHOWTAG* field in the VSI_TSR register of the VSI is posted to the *L2TAG1* field. The first tag is reported in the *L2TAG2* field. 1b = The above L2 tags are switched between the *L2TAG1* and *L2TAG2* fields.
HSPLIT_0	4	120	123	Static	HSPLIT_0	**Header Split 0** Header Split 0 control as described in Table 10-14.
HSPLIT_1	2	124	125	Static	HSPLIT_1	**Header Split 1** Header Split 1 control as described in Table 10-15.
Reserved	1	126	126	Static	0b	Reserved. Must be set to 0b.
SHOWIV	1	127	127	Static	SHOWIV	**Show Inner VLAN** The VLAN in the inner L2 header is stripped to the receive descriptor if enabled by this flag.
Reserved	46	128	173	Static	0x0	Reserved.
RXMAX	14	174	187	Static	RXMAX	**Max Packet Size** Max packet size for this queue defined in byte units. The *RXMAX* parameter defines the whole packet size starting at the L2 header up to and including the Ethernet CRC. The *RXMAX* must not be set to a larger value than 5 x *DBUFF* (since receive packet must never span on more than five buffers). Received packets larger than *RXMAX* are dropped and counted by the GLV_REPC counter of the VSI. ***Note:*** Packets larger than MFS defined per MAC are marked with l2_mac_err
Reserved	5	188	192	Static	0x0	Reserved. Must be set to zeros.
TPHRDesc	1	193	193	Static	TPHRDesc	**Read Descriptor TPH Enable** Descriptor fetch.
TPHWDesc	1	194	194	Static	TPHWDesc	**Write Descriptor TPH Enable** Descriptor write-back.
TPHData	1	195	195	Static	TPHData	**Packet Data TPH Enable**

Table 10-12. LAN Rx-Queue Context (256 Bits = 32 Bytes) [continued]

Alias	Width (bits)	LS Bit	MS Bit	Type	SW Init	Description
TPHHead	1	196	196	Static	TPHHead	**Packet Header TPH Enable**
Reserved	1	197	197	Static	0b	Reserved. Must be set to 0b.
LRXQTRESH	3	198	200	Static	LRXQTRESH	**Low Receive Queue Threshold** Defined in 64 descriptors units. When the number of free descriptors (defined by *TAIL* minus *HEAD*) "goes" below the *LRXQTRESH*, an immediate interrupt is triggered.
Reserved	55	201	255	Static	0x0...01	Reserved.

10.4.3.6.1.2 Receive Queue Context - Dynamic Section

This queue context section includes the fields frequently accessed by software. This section is organized as a set of two separate vectors, each of them indexed separately per Rx-Queue ID.

- Interrupt related registers QINT_RQCTL[n], where "n" is the queue index in the absolute Rx-Queue space.
- The *TAIL* pointer in the QRX_TAIL[n] registers for the PF and for the VFs, where "n" is the queue index of the function (PF or VF).

10.4.3.6.2 Miscellaneous Receive Queue Context

This section lists the queue context parameters that are used by the RCB block.

- Queue enablement flags in the QRX_CTRL[n] registers, where "n" is the queue index in the absolute Rx-Queue space. Queue enable and disable flow by the PF are explained in Section 10.4.3.1.
- Two additional bits to select and observe drop/no-drop traffic class behavior.
  - One bit enables a queue associated with a no-drop TC to start dropping received packets after a configurable timeout.
  - The other bit is a status bit indicating current state.

10.4.4 Stateless Receive Offloads

10.4.4.1 Strip Ethernet CRC Bytes

See Section 7.12.1.2.

10.4.4.2 Header Split

This feature consists of splitting a received packet into two separate regions based on the packet content. Splitting is usually between the packet header that can be posted to a dedicated buffer and the packet payload that can be posted to a different buffer (or multiple buffers). The sizes of these buffers are defined by the *DBUFF* and *HBUFF* parameters in the receive queue context. This kind of splitting is useful when different buffer allocation rules apply to these buffers, or there are different rules for TPH enablement.

Header split is enabled per receive queue by the *DTYPE*, *HSPLIT_0*, and *HSPLIT_1* fields in the receive queue context, as described in Table 10-13, Table 10-14, and Table 10-15, and illustrated in Figure 10-6.

Note: Figure 10-6 cannot cover all supported packet formats. but rather the most common cases that emphasize the header split functionality. For a complete set of packet formats supported by the default NVM settings, see the parse graph in Appendix A.2.

Split between the header buffer and the payload buffers and the status reporting is detailed in Table 10-16. The physical pointers to the header and payload buffers are defined in the receive descriptor.

Some rules regarding header split:

- A packet that has a MAC error (reported by the *RXE* flag) is posted as a whole to the packet buffers with no split.

  Note: Posting such packets to the host is enabled on in "Save Bad Frame" mode (enabled by the *SBP* flag in the PRT_SBPVSI register).

- For tunneled packets, the rules defined by *HSPLIT_1* parameter take precedence over those ones defined by the *HSPLIT_0* parameter. This means that if any flag in *HSPLIT_1* is enabled and the packet matches that setting, the packet is split according to *HSPLIT_1* regardless of *HSPLIT_0* settings.
- In each individual register (*HSPLIT_1* or *HSPLIT_0*), the packets are split according to the lowest matched entry in the tables below. If both *HSPLIT_0* and *HSPLIT_1* are set to zero, *DTYPE* in the receive queue context must be set to 00b. Otherwise (any header split is enabled by these registers), the *DTYPE* must be set to one of the header split options: 01b or 10b (explained below).
- If the packet is posted to multiple descriptors, only the header buffer of the first one is used.
- The packet header cannot span across multiple buffers. If the header buffer is smaller than the received header, the header is posted together with the packet payload. See Table 10-16.
- The header of a fragmented IP packet is defined up to including the IP header regardless if the fragment includes the L4 header. For IPv6 header, the IP header is defined up to and including the fragmented extension header.
- When a packet is replicated to multiple receive queues, the packet can be split differently on these queues according to their settings.
- Header split is supported for packets received from the network as well as local VM-to-VM traffic.

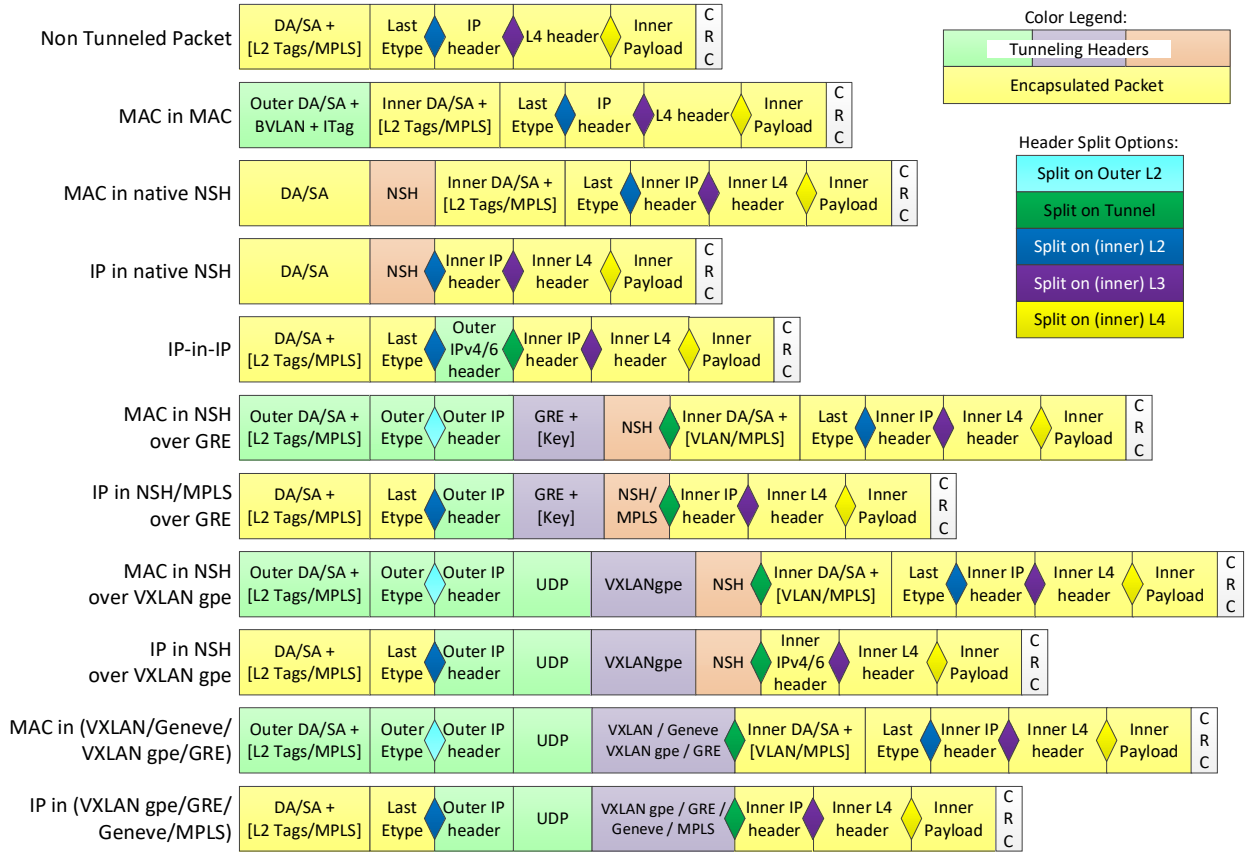


Figure 10-6. Header Split Option by Packet Formats

Table 10-13. Header Split Modes Defined by the DTYPE Field

DTYPE	Functionality
00b	**Single Buffer Descriptors** No header split mode.
01b	**Header Split Descriptors** Header split is enabled for packets that the hardware identifies their headers as enabled by the *HSPLIT* field. The packet content up to and including the most inner header enabled by the *HSPLIT* is posted to the header buffer. The rest of the packet is posted to the payload buffer. See Table 10-16 for rules of the header and payload buffers usage.
10b	**Split Always** Header split is always enabled regardless of the *HSPLIT* setting. If the packet header is identified as defined by the *HSPLIT*, the packet content up to and including the most inner header enabled by the *HSPLIT* is posted to the header buffer. See Table 10-16 for rules of the header and payload buffers usage.
11b	Reserved.

Table 10-14. Split Headers Enabled by the HSPLIT_0 Field

HSPLIT_0	Functionality
0000b	No Header are enabled by *HSPLIT_0*.
xxx1b	Enable split after L2 header. In case of L2 tunneling it is the second (inner) L2 header.
xx1xb	Enable split after the IP header. • In case of tunneling it is the second IP header. • In case of option/extension IP headers the split is after these headers. ***Note:*** If the tunneled pkt doesn't have a tunneled IP header, then split is done on the payload
x1xxb	Enable split after the UDP and TCP header (in case of UDP tunneling it is the second UDP header).
1xxxb	Enable split after the SCTP header.

Table 10-15. Split Headers Enabled by the HSPLIT_1 Field

HSPLIT_1	Functionality
00b	No split on tunneling headers.
x1b	Enable split after the outer (tunneling) L2 header. ***Note:*** Split is enabled only if inner L2 header exists.
1xb	Enable split on the entire tunneling header as follows: • Non-tunneled packet — No impact. • IP in any tunneling — Enable split before the inner IP header. It includes IP in IP, IP in GRE, IP in NSH or MPLS (w/wo prior headers), IP in any UDP tunneling. • MAC in any tunneling — Enable split before the inner MAC header. It includes MAC in any UDP tunneling or MAC in GRE.

Table 10-16. Header Split versus Packet and Buffer Sizes

DTYPE	Condition	Header and Payload DMA	SPH	HBO	PKTL[1]	HSRL[1]	Comments
00b (Single Buffer)	None.	Post the whole packet to the packet buffer(s)	0	0	Size of the whole packet	0x0	
01b (Header Split)	Header is not identified	Post the whole packet to the packet buffer(s)	0	0	Size of the whole packet	0x0	
	Header size ≤ HBUFF	Post header to header buffer and payload to the packet buffer(s)	1	0	Size of the packet payload	Header size	Packet Length calculation includes the Tags and CRC, which are striped later.
	Header size > HBUFF	Post the whole packet to the packet buffer(s)	1	1	Size of the whole packet	Header size	

Table 10-16. Header Split versus Packet and Buffer Sizes [continued]

DTYPE	Condition	Header and Payload DMA	SPH	HBO	PKTL[1]	HSRL[1]	Comments
10b (Always Split)	Packet length ≤ HBUFF	Post the whole packet to the header buffer(s)	0	0	0x0	Packet length	Packet Length calculation does not include the striped Tags and CRC.
	Header is not identified and packet length > HBUFF	Post the whole packet to the header buffer + packet buffer(s)	0	0	Size of the whole packet minus HBUFF	HBUFF	Packet Length calculation includes the Tags and CRC, which are striped later.
	Header size ≤ HBUFF	Post header to the header buffer and the payload to the packet buffer(s)	1	0	Size of the packet payload	Header size	Packet Length calculation does not include the striped Tags and CRC.
	Header size > HBUFF	Post the whole packet to the header + packet buffer(s)	1	1	Size of the whole packet minus HBUFF	Header size	

1. If the data posted to the packet buffer is larger than PKTL, multiple buffers (descriptors) are used.
   - All buffers but the last one are full (PKTL = DBUFF) while the last one contains the rest of the data.
   - Only the header buffer of the first descriptor is used.

10.4.4.3 Receive L3 and L4 Integrity Check Offload

The E810 offloads the following L3 and L4 integrity checks: IPv4 header(s) checksum, Outer UDP checksum, Inner TCP or UDP checksum, and SCTP CRC integrity (see Table 10-17). The E810 identifies the packet type and then checks the matched integrity scheme. The identified packet type is reported on the *PTYPE* field in the receive descriptor. Processing indication of the L3 and L4 headers is reported on the *L3L4P* flag in the receive descriptor. Potential IPv4 checksum error, L4 integrity error, and outer IPv4/UDP checksum error are reported by the IPE, L4E, and the EIPE error flags in the legacy receive descriptor, respectively. In the advanced receive descriptor it is reported in the flexible error flags.

Note: Outer UDP checksum is supported for all supported encapsulation over UDP (VXLAN, Geneve, and VXLAN-GPE).

Following are some rules for integrity check offload. If the following rules are not met, integrity offload is not provided and the *L3L4P* is not set.

- IPv4 header is assumed to be at least 20 bytes long (the length of the basic header).
- IPv4 headers can have any IP option headers that fit within the maximum header size (60 bytes).
- IPv6 support — The pseudo header for the L4 checksum takes into account the addresses in the IPv6 header, ignoring the optional extension headers. Packets with Routing Header type 2 and Destination Options Header with Home Address option contain an alternative IP Address in the extension header. Therefore, checksum calculation for such packets most probably results in an erroneous value. The E810 indicates the existence of a Destination Options Header or a Routing Header in the *IPV6EXADD* bit of the Rx-Descriptor. Software can then do one of the following:
  - — Ignore the checksum done by the device.
  - — Parse the extension header and identifying if it contains an IP Address, then ignore the checksum done by the device only in this case.

- Fragmented packets — The E810 parses fragmented receive packets up to and including the IP header (for IPv4) or up to and including the fragmentation extension header (for IPv6):
  - L4 checksum offload is not supported for IPv6 fragmented packets and the *L3L4P* flag in the receive descriptor is not set.
  - Fragmented IPv4 packet is offloaded up to including the IP header.
- TCP header is assumed to be at least 20 bytes long (the length of the basic header).
- The TCP header can have any option headers that fit within the maximum header size (60 bytes).
- VM-to-VM loopback traffic is processed by the hardware for L3/L4 integrity check as any other packet received from the network.

Table 10-17 lists all supported packet formats and the processed integrity. The table uses the following notations:

- IP is a generic term for IPv4 header or IPv6 header. The IPv4 header can have IP option headers and the IPv6 header can have IPv6 extension headers.
- L4 is a generic term for UDP, TCP, or SCTP headers.
- IP checksum is meaningful only for IPv4.
- Checksum is a generic term for UDP and TCP checksum, as well as SCTP CRC integrity.
- Zero UDP checksum — Zero UDP checksum for IPv4 packet is treated as no checksum and is reported by the hardware as "no error" and "done". Zero UDP checksum for IPv6 packet is illegal and is reported by the hardware as L4 checksum error.

Table 10-17. Integrity Offload Check for Receive Packet Types

Packet Type	Supported Integrity Offload	Reported L3L4P
IP -> [data / Unknown / fragmented]	IP checksum offload.	L3L4P is set if any header in the packet is checked for integrity. Note that IP checksum offload is true only for IPv4 header.
IP -> L4	IP and L4 checksum offload.	
IP -> IP -> [data / Unknown / fragmented]	2 x IP checksum offload.	
IP -> IP -> L4	2 x IP and L4 checksum offload.	
IP -> [tunnel header] -> IP -> data / Unknown / fragmented	Only 2 x IP checksum offload and tunneling UDP checksum (for UDP tunneling).	
IP -> [tunnel header] -> IP -> L4	2 x IP checksum, tunneling UDP checksum, and L4 checksum offload.[1]	
IP -> [tunnel header] -> data and IP -> [tunnel header] -> MAC -> data	IP checksum and tunneling UDP checksum offload.	
IP -> [tunnel header] -> MAC -> IP -> data	2 x IP checksum (relevant only for IPv4) and tunneling UDP checksum offload.	
IP -> [tunnel header] -> MAC -> IP -> L4	2 x IP, tunneling UDP, and L4 checksum offload.[1]	

1. The L4 checksum offload relates to the inner header:
   - For UDP or TCP protocols, the hardware calculates the expected checksum including the pseudo IP header.
   - For SCTP protocol, the hardware calculates the expected SCTP CRC.

10.5 LAN Transmit Data-Path

10.5.1 LAN Transmit Introduction

The following sections describe the initialization, configuration, data handling, disable and reset flows related to the LAN transmit functionality. This section covers the following major topics:

- A review of the transmit packet data stored in system memory, and the various assorted queue structures managed by software and used by hardware to perform data transmission.
- A discussion about the intended usage models, and the application of various features to these usage models.
- Initialization and configuration flows.
- Various packet transmission flows.
- Queue disable and reset flows.
- Error handling and additional considerations for software.

10.5.2 Transmit Packets in System Memory

Transmit packets are possibly multiple data non-contiguous buffers in host memory posted by software to hardware for transmission using descriptors (16-byte structures described in Section 10.5.3.2). These descriptors include pointer and length pairs to the data buffers, as well as control fields for the transmit data processing. In some cases, additional control parameters that cannot fit within the data descriptors are needed to process the packet(s). In this case, additional context descriptor(s) are posted by software to hardware prior to posting the data descriptors. A few common examples for context descriptor(s) are transmit segmentation (TSO) and Flow Director (FD) filter programming.

Figure 10-7 shows an example of a transmit packet in host memory composed of two buffers (header buffer and payload buffer), indicated by two matched "data descriptors" and an optional context descriptor.

The following rules are related to the transmit packet in host memory:

- The total size of a single packet in host memory must be at least 17 bytes and up to the "Max Frame Size" of the port as configured by the Set MAC Config admin command.
  - Packets outside this range are considered malicious. The respective queue is stopped and an interrupt is issued to the PF. The relevant event is "Bad Single Send size".
  - This rule applies for single packet send as well as any packet within a transmit segmentation (TSO).
- A single transmit packet can span up to eight buffers (up to eight data descriptors per packet including both the header and payload buffers).
  - When a packet spans on multiple buffers, all the descriptor of that packet must be filled similarly.
- The total number of data descriptors for the whole TSO (explained later on in this chapter) is considered unlimited (Limited only by Tx-Queue length) as long as each segment within the TSO obeys the previous rule (up to eight data descriptors per segment for both the TSO header and the segment payload buffers).

- If a packet or TSO spans on multiple transmit data descriptors, the fields in all the data descriptors must be valid.
- The TSO message header should not span on more than three buffers (Max three descriptors).

Enhanced Host-interface modes as explained in Section 10.5.6.2 places additional restrictions on the maximum number of descriptors, maximum number of descriptors per quanta, maximum commands per quanta, and the offload commands that can be used.

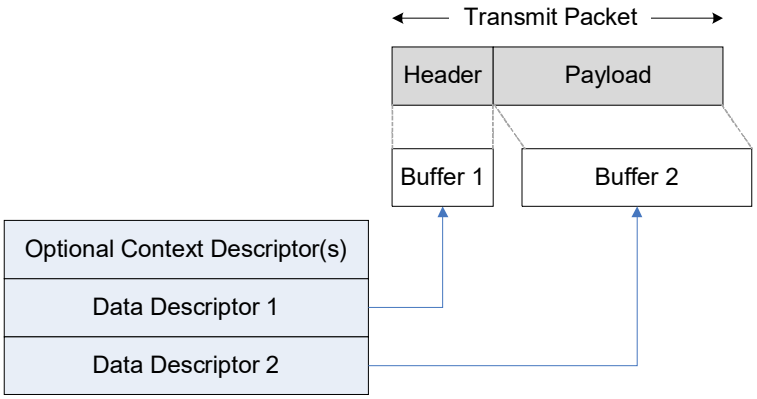


Figure 10-7. Transmit Packet in System Memory (Example Using Two Buffers)

10.5.3 Descriptors and Doorbells

The software prepares structures for transmission in system memory indicated to hardware by a list of consecutive descriptors. These descriptors are organized in a contiguous memory handled as a cyclic queue, which is also called a Transmit Descriptor Ring (TDR). Descriptors are initialized by software, and posted to hardware for processing via a write to a doorbell register. The E810 supports up to 16K transmit queues allocated to PFs and VFs.

Transmit queue state and behavior is initialized by software, which programs a set of parameters collectively called the *queue context*. The main parameters are the queue pointers presented in Figure 10-8. The queue context includes additional parameters that define the queue functionality as detailed in Section 10.5.5.2. The context parameters are kept in an internal memory within hardware.

The software interface to the queue for initialization, nominal operation, and queue disable is described in Section 10.5.5.1. The E810 has additional global parameters for the whole device or per function parameters that affect multiple queues.

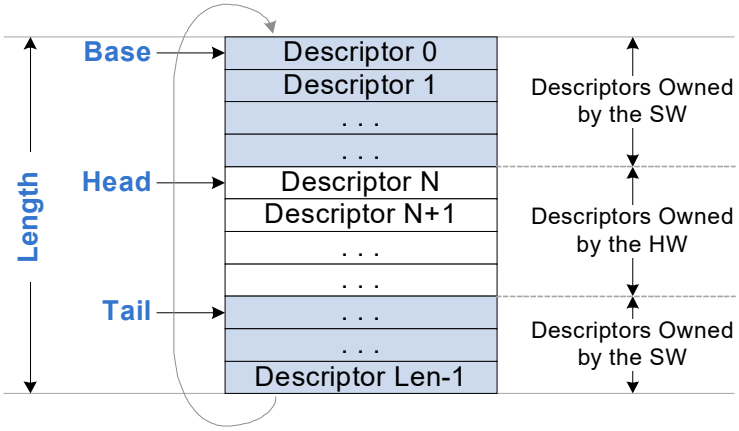


Figure 10-8. Transmit Descriptor Ring Structure

When software posts a packet for transmission, it can add some specific rules and commands per the transmitted packet. This is done by adding extra descriptors on top of the Data descriptors that point to the transmitted packet.

The following subsections describe the various descriptor types.

10.5.3.1 General Descriptors

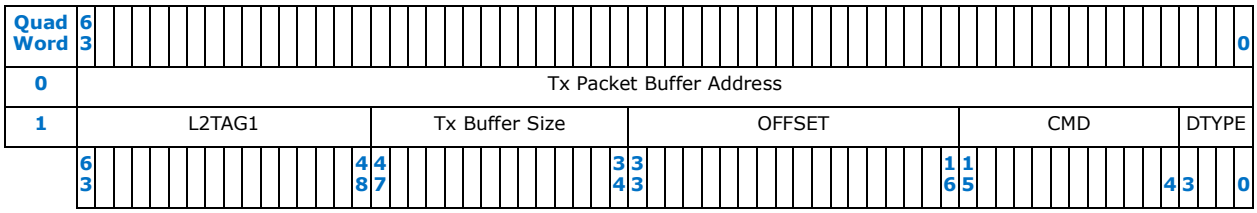
Table 10-18. LAN Descriptor Types

Descriptor Type	DTYPE Value	Description	Section Reference
NOP (optional)	0x1	NOP descriptors can be used to align descriptors to cache lines (optional usage).	10.5.3.1.2
LAN Transmit Context	0x1	Used as a companion to the Transmit Data descriptor to provide more information on the packet.	10.5.3.2
FD Filter Programming	0x8	Used to program Flow Director filters.	10.5.3.3
Transmit Data	0x0	Regular data descriptor used to send LAN packets	10.5.3.1.1

Notes:

- For all descriptors, fields indicated as "RSV" (reserved) should be set to zero by the software at programming time.
- Descriptor type 0x8 is excluded in a single packet transmit.
- Software can use multiple descriptors for a single packet transmission. For example, a transmission of a single LSO message. For this type of packet, software is required to post a Context Descriptor (0x1) followed by one or more Data Descriptors (0x0). In any case multiple descriptors are used for a transmitted packet, software must place them into the Tx-Queue in the order they are documented in Table 10-18.

10.5.3.1.1 Transmit Data Descriptor



Descriptor Type - DTYPE (Quad Word 1, Bits 0:3, 4 bits)

0x0 stands for a Transmit Data Descriptor.

Command Field - CMD (Quad Word 1, Bits 4:15, 12 bits)

Bit(s)	Name	Description
0	EOP	End of Packet. The EOP flag is set in the last descriptor of a packet or TSO.
1	RS	Report Status. When set, the hardware reports the DMA completion of the transmit descriptor and its data buffer. Completion is reported by Completion Queue or by descriptor write-back as configured by the *HEAD_WBEN* flag in the transmit context. When it is reported by descriptor write-back, the *DTYPE* field is set to 0xF and the *RS* flag is set. The *RS* flag can be set only on a last Transmit Data Descriptor of a packet or last Transmit Data Descriptor of a TSO.
2	Reserved	Reserved. Must be set to 1b.
3	IL2TAG1	Insert an L2 tag from the L2TAG1 field in this descriptor. For MAC in UDP or MAC in GRE tunneling, the *L2TAG1* field is relevant to the outer L2 header. See *IL2TAG_IL2H* for VLAN tag insertion to the inner L2 header. The type of the Tag is defined by the *L2TAG1INSERTID* field in the VSI_L2TAGSTXVALID register of the VSI. The structure of this tag is defined by the matched GL_SWT_L2TAG*xxx* registers. If the type of the L2 Tag requires more than two variable bytes, additional bytes are taken from the *L2TAG2* field while the *L2TAG1* is first on the wire. In this case, the *IL2TAG2* flag must be cleared.
4	DUMMY	When the *DUMMY* flag is set, the packet is not transmitted (internal and external). The *DUMMY* indication can be useful for context programming purposes: FD filters. Note that when using the *DUMMY* option, the packet does not have to have a correct checksum. Software should set all the fields in the data descriptor and context descriptors describing the packet structure as it does for nominal packets.
5:6	IIPT	The IP header type and its offload. In case of tunneling, the IIPT relates to the inner IP header. See also *EIPT* field for the outer (External) IP header offload. 00b = Non-IP packet or packet type is not defined by software. 01b = IPv6 packet. 10b = IPv4 packet with no IP checksum offload. 11b = IPv4 packet with IP checksum offload. For an IPv4 TSO message, this field must be set to 11b.
7	Reserved	Reserved.
8:9	L4T	L4T is the L4 packet type. 00b = Unknown/Fragmented packet 01b = TCP 10b = SCTP 11b = UDP When the *L4T* is set to values other than 00b, the *L4LEN* must be defined as well. When set to UDP or TCP, the hardware inserts the L4 checksum and when set to SCTP the hardware inserts the L4 CRC. ***Note:*** Requesting SCTP CRC/TCP or UDP offload for a packet that was padded by Software results in wrong SCTP CRC.

Bit(s)	Name	Description
10	RE	Report Event. When set, the hardware reports the DMA completion of the transmit descriptor and its data buffer. Completion is reported by Completion Queue or descriptor write-back as configured by the queue context parameters. When it is reported by descriptor write-back, the *DTYPE* field is set to 0xF and the *RE* flag is set. The *RE* bit setting must follow the same rules as the *RS* bit, described above.
11	Reserved	Reserved.

Header Offset Parameters - OFFSET (Quad Word 1, Bits 16:33, 18 bits)

Bit(s)	Name	Description
0:6	MACLEN	MAC Header Length defined in Words. In case of a tunnel packet, *MACLEN* defines the outer L2 header. *MACLEN* defines the L2 header length up to including the EtherType. If L2 Tag(s) are provided in the data buffers, they are included in *MACLEN*.
7:13	IPLEN	IP header length (including IP optional/extended headers) in the Tx buffer defined in DWords. In case of a tunnel packet, *IPLEN* defines the most inner IP header length. If the *IIPT* flags cleared, this field should be set to zero.
14:17	L4LEN	*L4LEN* is the L4 header length in the Tx buffer defined in DWords. In case of a tunnel packet, *L4LEN* defines the most inner L4 header length. *L4LEN* should obey the following rules: When the *L4T* field is set to 00b, *L4LEN* must be set to zero. Otherwise, it should be set to 8/12 for UDP/SCTP, respectively, and should be greater than or equal to 20 for TCP.

L2 Tag 1 - L2TAG1 (Quad Word 1, Bits 48:63, 16 bits)

A 16-bit tag to be inserted into the packet if the *IL2TAG1* flag is set. If *IL2TAG1* is cleared, *L2TAG1* should be set to zero by software.

Tx Buffer Size (Quad Word 1, Bits 34:47, 14 bits)

Bit(s)	Name	Description
0:13	BSIZE	Buffer size in byte units from 1 byte up to 16 KB minus 1.

Tx Packet Buffer Address (Quad Word 0, Bits 0:63, 64 bits)

Bit(s)	Name	Description
0:63	BADDR	Buffer address in byte granularity.

10.5.3.1.2 NOP Descriptor

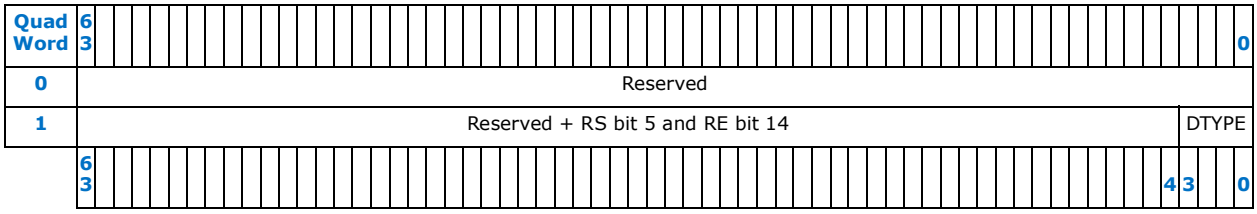
A NOP descriptor can be used for padding if software needs to align descriptors for cache alignment or other performance reasons. A NOP descriptor does not cause any activity other than processing of the descriptor. Most importantly, NOP descriptors do not generate write-backs to indicate the hardware has processed the descriptor.

NOP descriptors are implemented by the "Null" setting of a context descriptor as follows: DTYPE should be set to 0x1 (LAN Context Descriptor) and all other fields should be cleared. Note that NOP descriptors are permitted only between packets.

NOP descriptors are reported as a drop event when the queue is configured with Completion Queue and Head Drop.

When the queue is configured for advanced host interface mode, NOP descriptor can be issued only as a piggyback at end of a valid packet in a single doorbell.

10.5.3.1.3 Transmit Descriptors Write-Back Format



Descriptor Type - DTYPE (Quad Word 1, Bits 0:3, 4 bits)

The hardware indicates a completed descriptor by setting the *DTYPE* field to a value of 0xF. The write-back status is the same for all transmit descriptors used for LAN traffic, filter programming.

The hardware reports this status in the following two cases:

- For descriptors with the *RS* and/or *RE* bit set (Report Status, Report Event) while the Tx-Queue is configured for descriptor write-back completion (*WB_MODE* field in the queue context is set to 0). See rules for setting the *RS* and/or *RE* flag in the transmit descriptors in Section 10.5.3.1.1.
- For the last descriptor of a command that was executed before an interrupt of the queue is initiated.

Completion Flags - RS and RE (Quad Word 1, Bit 5 and Bit 14, respectively)

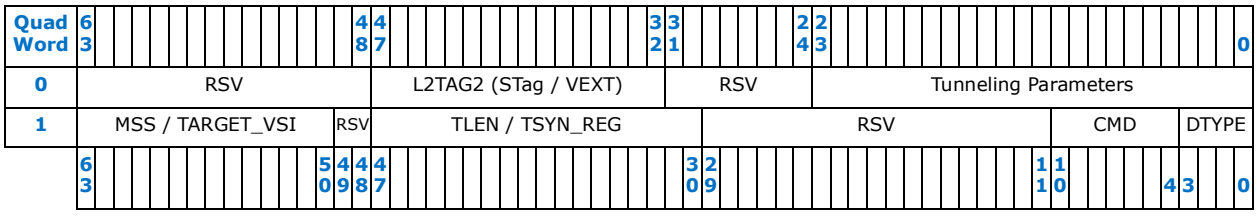
RS bit is set if it was active by software.

RE bit is set if it was active by software.

Note: The bits locations are as they are in the data descriptor.

10.5.3.2 LAN Transmit Context Descriptors

A context descriptor can contain some additional setting options for a single packet or TSO defined by the Data descriptor(s) that follows. Following the transmission of the packet or TSO, the context provided by this descriptor is “expired”.



Descriptor Type - DTYPE (Quad Word 1, Bits 0:3, 4 bits)

0x1 stands for a LAN Context Descriptor.

Command Field - CMD (Quad Word 1, Bits 4:10, 7 bits)

Bit(s)	Name	Description
0	TSO	Transmit Segmentation Offload is activated when the *TSO* flag is set. Applicable for both TCP and UDP packets.
1	TSYN	1588 Timestamp. When set, the hardware samples the packet transmission time in one of PRTTSYN_TXTIME[64] registers. The TSYN flag is processed by the device only if TimeSync is enabled by the *TSYNENA* flag in the transmit queue context. ***Note:*** The *TSYN* flag is aimed only for a single send with no TSO. When *TSYN* is set, the *TSYN_REG* field must be initialized as well.
2	IL2TAG2	Insert the L2 tag from the L2TAG2 field in this descriptor. For MAC in UDP or MAC in GRE tunneling, the *L2TAG2* field is relevant to the outer L2 header. See *IL2TAG_IL2H* for VLAN tag insertion to the inner L2 header. The type of the tag is defined by the *L2TAG2INSERTID* field in the VSI_L2TAGSTXVALID register of the VSI. The structure of this tag is defined by the matched GL_SWT_L2TAG*xxx* registers. See also *IL2TAG1* in the data descriptor for 32-bit tag handling.
3	IL2TAG_IL2H	Insert VLAN to the inner L2 header from the L2TAG2 field in this descriptor. The VLAN is added after the header defined by *L4TUNLEN*. The *IL2TAG2* and *IL2TAG_IL2H* bits are mutually exclusive and the software can set only one of them. The *IL2TAG_IL2H* can be set to 1b only for L4 tunneled packets with non-zero *L4TUNLEN*. Inserting the VLAN in the inner L2 header, the hardware also updates the IP total length field in the outer IP header. It is done for Single-Send as well as for TSO.
4:5	SWTCH	Switch Control Tag (Can be set to non-zero only by control VSI as programmed by the destination override" flag per VSI): 00b = No Switch Control Tag. The packet is routed according to the hardware filters. 01b = Uplink packet. The packet is transmitted to the network bypassing hardware filters. 10b = Local packet. The packet is transmitted only to local VSIs according to the hardware filters. 11b = Target VSI. The packet is transmitted to a specific VSI defined in the MSS/VSI field in this descriptor. TSO is mutually exclusive with this option.
6	Reserved	Reserved.

Segmentation Parameters / Switching Parameters (Quad Word 1, Bits 30:47; Bits 50:63, 32 bits)

Bit(s)	Name	Description
30:47	TLEN / TSYN_REG	For a TSO message: TSO Total Length. This field defines the L4 payload bytes that should be segmented. Note that the sum of all transmit buffer sizes of the TSO should match exactly the *TLEN* plus the TSO header size in host memory. If the *TSO* flag is cleared, the *TLEN* should be set to zero by software. If the *TSO* flag is set, the *TLEN* must be set by software to a value larger than zero. For a TSYN packet: Bits 30:35 = Contain the index for TSYN_REG[64] register, which is used for this timestamp sampling. Bits 36:47 = Reserved. Must be set to zero by software.
50:63	MSS / TARGET_VSI	When the TSO flag is set, this field functions as a MSS. When the *SWTCH* field is set to 11b, this field functions as a TARGET_VSI. If the *TSO* flag is set, the *SWTCH* field should not be set to 11b and vice-versa. If both the *TSO* flag is cleared and the *SWTCH* field is not equal to 11b, then this field should be set to zero. When the TSO flag is set, the MSS field defines the Maximum Segment Size of the packet's payload in the TSO (excluding the L2, L3 and L4 headers). In case of tunneling, the MSS relates to the inner payload. The M.SS should not be set to a lower value than 88. It also must follow the rule declared in field When the *SWTCH* field equals to 11b, it is the destination VSI of the packet. This option is valid only for control VSIs on which the "Allow destination override" flag is set.

Tunneling Parameters (Quad Word 0, Bits 0:23, 24 bits)

Bit(s)	Name	Description
0:1	EIPT	The External (outer) IP header type and its offload: 00b = No External IP header. 01b = External IPv6. 10b = External IPv4 with no checksum offload. 11b = External IPv4 with checksum offload. For an IPv4 TSO message, this field must be set to 11b.
2:8	EIPLEN	External (outer) IP header length (including IP optional/extended headers) defined in DWords. When the packet has no outer IP header (EIPT equals to zero), this field must be set to zero.
9:10	L4TUNT	L4 Tunneling Type (GRE header / VXLAN header) indication: 00b = No UDP / GRE tunneling (field must be set to zero if EIPT equals to zero). 01b = UDP tunneling header (Any UDP tunneling, VXLAN and Geneve). 10b = GRE tunneling header. 11b = Reserved.
11	Reserved	Reserved. Must be set to 0b.
12:18	L4TUNLEN	L4 Tunneling Length (GRE header / VXLAN header) defined in Words (field must be set to zero if *L4TUNT* equals to zero). • If the tunneling header includes proprietary content it should be included as well. • For IP in GRE it should be set to the length of the GRE header. • For MAC in GRE or MAC in UDP it should be set to the length of the GRE or UDP headers plus the inner MAC up to including its last EtherType. ***Note:*** This field represents the length of data provided in host memory buffers. If the *L4TUNT* is cleared, this field must be set to zero. If MPLS labels exists, *L4TUNLEN* should include them as well.
19:22	DECTTL	Decrement TTL in the inner IP header by *DECTTL* and drop the packet if the original TTL was not greater than *DECTTL*. If the *EIPT* is cleared, this field must be set to zero. ***Note:*** If software attempts to transmit a packet with a value of zero in the TTL header field, the packet is dropped regardless the *DECTTL* value.
23	L4T_CS	Calculate the tunneling UDP checksum. Must be set only if *L4TUNT* = 01b and *EIPT* is not zero.

L2 Tag 2 - L2TAG2 (Quad Word 0, Bits 32:47, 16 bits)

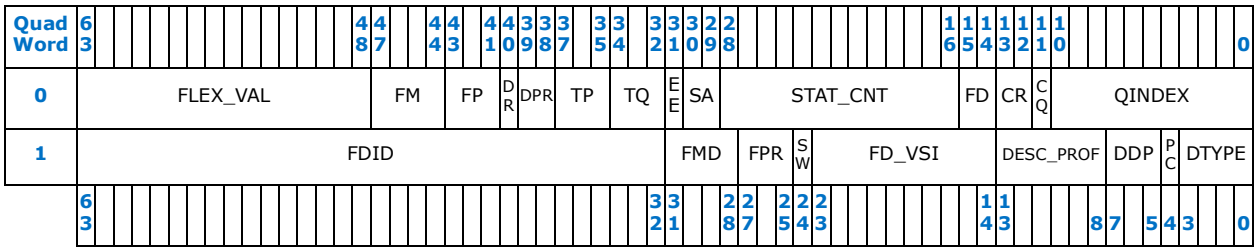
A 16-bit tag to be inserted to the packet if the *IL2TAG2* flag is set, or if *IL2TAG1* is set while L2 Tag 1 include four variable bytes, or if the *IL2TAG_IL2H* flag is set.

If the above conditions are not met, the *L2TAG2* should be set to zero by software.

RSV

Reserved bits that must be set to zero.

10.5.3.3 FD Filter Programming Descriptor



Quad Word 0

Bit(s)	Name	Field Size	Description
0:10	QINDEX	11	Destination queue index or region of queues within the VSI space. It is controlled by the *ToQueue* and *ToQueue_PRIO* parameters. The *QINDEX* must be set to zero if the *ToQueue_PRIO* equals zero.
11	COMP_QUEUE (CQ)	1	The receive queue of the programming VSI for the completion status is selected by the *COMP_QUEUE* flag: 0b = Receive queue zero. 1b = The Rx-Queue defined by the *COMP_QINDX* field in VSIQF_FD_DFLT.
12:13	COMP_REPORT (CR)	2	*COMP_REPORT* controls the completion status reported to software, as follows: 00b = No completion status is reported to software. 01b = Completion status is reported to software in the case of failed programming. 10b = Completion status is reported to software for any programming request. 11b = Reserved.
14:15	FD_SPACE (FD)	2	At programming time, software can select between best-effort and guaranteed spaces in the FD table using the *FD_SPACE* field: 00b = The FD filter is programmed only on the guaranteed space of the function. 01b = The FD filter is programmed only on the best effort space of the function. 10b = The FD filter programming starts with the guaranteed space of the function. If no space, programming is made on the best effort space of the function. 11b = The FD filter programming starts with the best effort space of the function. If no space, programming is made on the guaranteed space of the function.
16:28	STAT_CNT	13	Statistic counter(s) index(es) associated with this filter as controlled by the *STAT_ENA* field. The statistic counters are implemented in 2 "banks" with 4K counters on each bank. The MS bit of the *STAT_CNT* selects the bank and the LS bits selects the counter(s) within the bank(s). If the FD filter uses a single counter, it can be selected on any bank. If the FD filter uses both packet and byte counters, it is the same counter index in both banks. In this case, the MS bit must be set to zero.
29:30	STAT_ENA (SA)	2	The *STAT_ENA* controls the functionality of the *STAT_CNT*, as follows: 00b = No statistic counter associated with the filter. The *STAT_CNT* must be set to zero. 01b = The *STAT_CNT* defines a statistic counter that counts packets. 10b = The *STAT_CNT* defines a statistic counter that counts bytes. 11b = The *STAT_CNT* defines 2 statistic counters that counts packets and bytes. Refer Section 7.10.8.1, "Statistic Counters" for counter indexing rules.
31	EVICT_ENA (EE)	1	Filter programmed with the *EVICT_ENA* setting can be auto-evicted by FIN/RST packets.
32:34	ToQueue (TQ)	3	The *ToQueue* parameters controls the functionality of the *QINDEX,* as follows: 0 = The filter defines a target queue equal to *QINDEX*. 1-7 = The filter defines a region of queues for the hash filter. The region base equals to *QINDEX*. and the region size equals to 2^(*ToQueue*).
35:37	ToQueue_PRIO (TP)	3	To Queue priority action by the FD filter. Priority = zero means no to Queue action.
38:39	DPU_Recipe (DPR)	2	Reserved for DPU Recipe option. Must be set to 00b.
40	DROP (DR)	1	The matched packet is dropped when the *DROP* flag is set.
41:43	FLEX_PRIO (FP)	3	Flexible action priority by the FD filter. Priority = zero means no to flexible action.
44:47	FLEX_MDID (FM)	4	Flexible action type. Valid values are 0 to 4.
48:63	FLEX_VAL	16	Flexible action by the FD filter. Must be set to zero if the *FLEX_PRIO* = 0.

Quad Word 1

Bit(s)	Name	Field Size	Description
0:3	DTYPE	4	0x8 for a FD filter programming descriptor.
4	PCMD (PC)	1	Filter Programming Command: 0b = Add filter.(*) 1b = Remove filter. (*) If the filter entry does not exist, the filter is added. If the filter already exists, the filter parameters are updated. Filter is candidate for update if all the fields that are used for the packet identification match: Packet tuples, VSI, PACKET_DIR.
5:7	DESC_PROF_PRIO (DPP)	3	Receive descriptor profile priority by the FD filter. Priority = zero means no descriptor profile by the FD filter.
8:13	DESC_PROF	6	Receive descriptor profile. See Section 7.6 for supported profiles. Must be set to zero if the *DESC_PROF_PRIO* equals zero.
14:23	FD_VSI	10	Expected VSI index of the packet during reception. Note that the VSI index might be assigned by the embedded switch or any other logic that overrides it before the packet is processed by the FD filter.
24	SWAP (SW)	1	When the SWAP flag is set, "source" and "destination" fields in the Tx packet used for the filter programming are swapped. See description of the GLQF_FDSWAP register in Section 7.10.4.2.
25:27	FDID_PRIO (FPR)	3	FDID priority. Priority = zero means no FDID by the filter.
28:31	FDID_MDID (FMD)	3	FDID action type. It should be set to 0x05 for FD filter ID. Valid values are 0 to 5.
32:63	FDID	32	The functionality of the FDID is defined by the *FDID_MDID* field. It must be set to zero if the *FDID_PRIO* equals zero.

10.5.4 LAN Transmit - Advanced Features

The E810 is a device designed to support the COMMS market usage models as well. On the Tx side, this includes the COMMS Tx-Scheduler detailed in Section 8.3, "Transmit Scheduling" and the Tx advanced host interface.

10.5.4.1 Advanced Mode Host Interface

The Advanced mode host interface includes some new types of services, as follows:

- **Many Tx-Queues**

  To address fine-grained QoS in Tx, the transmits to many targets are controlled and paced separately. This is done by supporting many Tx-Queues to allow the fine distinction between targets and class of services. The E810 supports 16K transmit queues. Each one of the Tx-Queues is individually controlled by the Tx-Scheduler.

- **No Locality Assumption for COMMS Queues**

  In Enterprise and Cloud usage models, the transmitted data comes from applications running on the host platform. Those applications can be part of the PFs, and also belong to VMs or VFs in a virtualized environment. Each function is allocated with set of Tx-Queues. The amount of resources associated with a function is derived from the amount of CPU cores the function runs, and the amount of TCs it uses. For the Enterprise and Cloud usage models, the assumption is that the E810 needs 2048 Tx-Queues to address all functions Tx needs. As mentioned above, for the COMMS usage models, many Tx-Queues are required (16K in the E810).

Another difference between the Enterprise/Cloud and the COMMS usage model is the locality. In the Enterprise/Cloud usage models, not all the VMs run simultaneously on the CPU core. Rather, they are swapped in and out. Only the VMs that actively run generate Tx traffic. Hence, the assumption is that simultaneously a limited amount of Tx-Queues (256 in the E810) are active. In COMMS usage models, a small amount of software entities run on the platform anyway. Each entity uses many queues for its QoS purposes.

The E810 manages internal descriptors cache to meet the Tx performance requirements while optimizing the PCIe bandwidth. Cache helps to improve performance under some locality assumption. This works for the Enterprise/Cloud usage models.

As explained in Section 10.5.4.3, the E810 supports three types of Tx-Queues. Some of them use internal cache to accommodate device's performance requirements. The pure COMMS host interface is architected to meet the required performance without using the descriptors caching. It uses the Richer doorbell format explained below and the quanta descriptors explained in Section 10.5.4.4.

10.5.4.2 Advanced Mode Host Interface New Terms and Entities

The Advanced mode host interface includes some new type of entities, as follows:

- **Richer Doorbell Format**

  Together with quanta queue, it enables accurate scheduling, as well as accurate usage of internal resources and caches. This way, the device operates fine-grained traffic management with high performance for 16K queues with no locality assumption.

  This adds some complexity to software flows. So, pure COMMS applications are not limited for 256 locality but pay by some complexity in the Host interface.

  The Advanced Mode Doorbell is detailed Section 10.5.4.6.

- **Tx-Queue Behavior Profiles**

  The Tx-Scheduler of the X710/XXV710/XL710 worked with one fixed quanta size (4 KB) over all Tx-Queues and RDMA QSets. In the E810, each Tx-Queue can be configured to operate in different quanta size. Some other features are also profiled per Tx-Queue.

  Each Tx-Queue is assigned with three types of profiles, Quanta Profile, Packet Shaping Profile, and Cache Profile. The Tx-Queue behaves according to the profile definitions it is associated with.

- **Quanta Profile**

  Each Tx-Queue is associated with one of 16 configured quanta profiles. The quanta profile includes the following parameters:

  — **Max Descriptors in a quanta:**

    - In Legacy queue, it defines the number of descriptors fetched when the queue is scheduled according this parameter.
    - In advanced mode for TSO segments, it defines how many descriptors are fetched for each TSO quanta.

  — **Max Commands in a quanta:**

    - Similar to the previous, in Legacy queue or in LSO enabled queue, related to internal structures of packet processing. Number of commands per packet is driven from number of descriptors represent the packet.

  - **Configured quanta size:**
    - Define the quanta size used by the legacy queues or in TSO.

  The Quanta Profile structure is detailed in Section 10.5.5.3.

- **Packet Shaping Profile**

  Used in Legacy Host Interface Queue (Queue type #1 detailed in Section 10.5.4.3). The E810 Tx-Scheduler can shape and schedule traffic based on bytes-per-second, or based on packets-per-second. This feature is configurable per each scheduling group in any scheduling layer. When the queue operates in legacy host interface mode, The Packet Shaper Profile provides a predictable number of packets per scheduled quanta. Like in the regular speculative scheduling, after the queue is scheduled and the quanta is processed, an update with the real amount of packets is sent to the Tx-Scheduler to keep all further scheduling decision accurate.

  The Number of Packets Profile structure is detailed in Section 10.5.5.5.

- **Cache Profile**

  Used to fine tune quanta descriptor caching between highest single queue performance and faster response to Head Drop. Described in Section 10.5.5.4.

- **Quanta Queue**

  Quanta Queue is explained in Section 10.5.4.4.

- **Completion Queue**

  Completion Queue is explained in Section 10.5.4.5.

- **Doorbell Queue**

  Doorbell Queue is explained in Section 10.5.4.7.

- **Head Drop**

  Two types of Head Drop are explained in Section 10.5.6.2 and in Section 10.5.6.3.

10.5.4.3 LAN Transmit Queue Modes

Each queue managed by the host interface operates in one of two modes - Legacy and Advanced Transmit. The Advanced Transmit mode enables software to tightly control the burstiness of the traffic from this queue, and to interleave transmissions between multiple data sources while limiting the burst size from any one given queue.

To assist the LAN transmit scheduler to control the burst size, software provides ongoing run-time data in the form of Quanta Descriptors posted to Quanta Descriptor Rings, as well as in the doorbell writes. These scheduled quanta include the number and overall size of data posted for transmission to a given queue. The device-based transmit scheduler uses this data to schedule and pace packets.

The E810 supports two Tx-Queue host interface modes:

- **Legacy Mode (X710/XXV710/XL710 backward compatible):**

  Exposed VF interfaces are binary compatible with prior generation products. This mode supports the usage model of migrating virtual machines utilizing SR-IOV between product generations of the same family (the X710/XXV710/XL710 and the E810).

  PF managed transmit queue configuration can be changed relative to the X710/XXV710/XL710, as the PF driver must be updated to support the E810.

- **Advanced Transmit Mode:**

  A higher granularity and controlled burst size for packet transmission scheduling.

Both Legacy and Advanced queues can transmit regular packet (SSO) and TSO messages. The configuration for Legacy versus Advance Host interface is per Tx-Queue. The hardware identifies three types of behavior in Tx-Queue modes:

- **Legacy Queue** — As described above. This queue operates in legacy interface and can transmit SSO and TSO messages.
- **Advanced Mode with TSO Support** — Tx-Queue in this operation mode uses the advanced host interface and enjoys the burst control for both SSO and TSO messages. This queue type does not support backward compatibility with previous device generations.
- **Advanced Mode with 'NO' TSO Support** — This queue uses the advanced host interface with no TSO support.

The E810 supports up to 16K Tx-Queues. Each one of these 16K queues can be configured to operate in one of three above modes of operation with some limits:

- The total amount of queues that are configured for mode 1 or mode 2 must not exceed 2K at any given time.
- The performance goals of Tx are calculated under the locality assumption where no more than 256 Tx-Queues that are configured for mode 1 or for mode 2 are active simultaneously. There is no locality restriction for queues of mode 3.

In this section, where the document refers to LSO-enabled queue, it can be a mode 1 or mode 2 queue.

10.5.4.4 Quanta Descriptor

Quanta descriptors are submitted on a physically adjacent but separate queue (the QDQ) from the associated transmit queue, with an equal number of descriptors defined as the transmit queue. Quanta descriptors themselves are eight bytes in length. Two types of quanta descriptors are supported:

- Single Send (SSO)
- Transmit Segmentation Offload (TSO).

During the descriptor posting flow, when software completes a quanta, software also writes a quanta descriptor to the QDQ associated with the transmit queue.

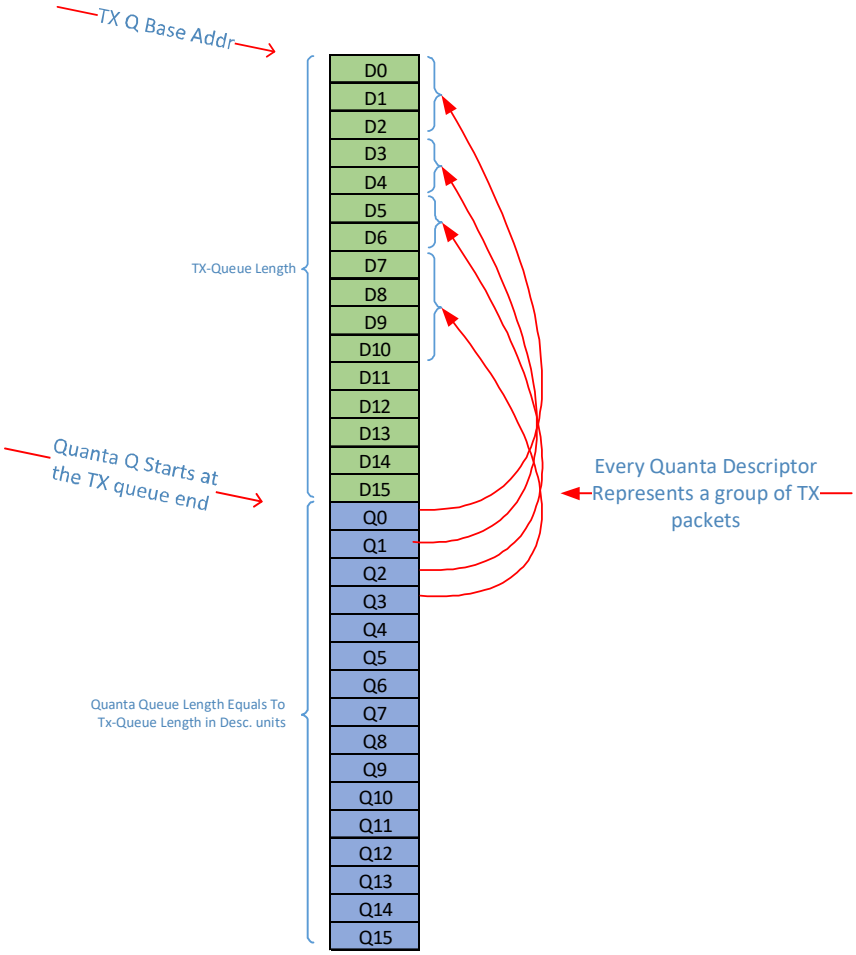


Figure 10-9. Quanta Descriptor and Data Descriptor

Every TSO message is considered as a standalone quanta in the QDQ, and the device can transmit the TSO in multiple quanta's, but software posts the TSO operation as a single standalone quanta descriptor. This means that any partially-filled quanta from previous SSOs are closed automatically by hardware before submitting the new quanta descriptor associated with the TSO. Furthermore, the quanta descriptor associated with the TSO is automatically closed and not aggregated with any later operation.

Table 10-19. SSO Quanta Descriptor Format

Field	Bit(s)	Description
Type	0	Quanta descriptor type. Set to 0b for Single Send.
Desc	6:1	Amount of data descriptors occupied by the quanta packets. Up to 63 descriptors per quanta.
Length	14:7	Data length in 64-bytes resolution - up to 16K-64 bytes.
Reserved	15	Reserved. Must be set to 0b.
Number of Packets	21:16	Number of packets incorporated in the quanta. A quanta can include 1-63 packets. Value of 0 is reserved for special cases.

Table 10-19. SSO Quanta Descriptor Format [continued]

Field	Bit(s)	Description
Reserved	31:22	Reserved. Must be set to zero.
Expire Timestamp	49:32	Expire timestamp.
Timestamp Drop Flag	50	If this bit is set to 1b, the device compares the *Expire Timestamp* field against the internal timestamp, and drops the quanta if needed.
Tx-Descriptor	63:51	Tx-Descriptor ID. Points to the first descriptor of the quanta. This data helps the device in the Head Drop implementation. If the Head Drop feature is not enabled for this queue (*drop enable* bit in queue context is cleared), this field is set to zeros.

Table 10-20. TSO Quanta Descriptor Format

Field	Bit(s)	Description
Type	0	Quanta descriptor type. Set to 1b for TSO.
Desc	13:1	Amount of descriptors occupied by the TSO message.
Length	31:14	Data length in bytes resolution - up to 256KB-1 bytes. This field defines the L4 payload bytes that should be segmented.
MSS	46:32	Maximum Segment Size (MSS) in bytes. The required maximal segmentation size. Up to 9.5KB.
Header Size	55:47	Header size. The size of the header in bytes.
Reserved	63:56	Reserved.

10.5.4.5 Completion Queue (CQ) Descriptor

After each transmit packet is fetched from host memory into device memory, or on packet transmission, a completion notification can be provided by the device. The E810 supports two completion methods

- **Descriptor write-back** — An existing feature.
- **Completion Queue** — A new feature in the E810.

Completion mode is configurable per Tx-Queue. When a Tx-Queue is configured for Completion Queue method, it is also associated with a Completion Queue.

Completion Notification is generated in three cases:

- Software marks transmit descriptors with either Report Status (RS) or Report Event (RE) request in the Tx-Descriptor.

  Note: In TSO transmits, *RS* or *RE* bit can be set only at the last descriptor of the TSO message.

- When a packet is dropped on transmit due to timestamp expiration or an explicit drop request from software, the E810 might or might not coalesce drop notifications in case of two contiguous drop events.
- When ITR expires and there are outstanding completions that have not yet been reported to software.

When a completion notification is targeted to a Completion Queue, it is not immediately written to the host memory. Each Completion Queue structure includes a 64-byte internal cache. The completion notifications are coalesced.

Completion notifications are written to host memory when:

- The internal cache is full with 16 notification (64 bytes).
- At the end of ITR expire flow, the internal completion cache is written to the host even if it is not full.
- When a Queue Disable marker or a VM reset marker is written to the Completion Queue, the internal completion cache is written to the host even if it is not full.

Table 10-21. Completion Queue Descriptor Format

Field	Bit(s)	Description
Tx Q ID / VM ID	13:0	Transmit Queue ID this completion refers to. Queue ID in the function space as configured in the Tx-Queue context field *qnum_in_func*. See Table 10-28 on page 1337. When completion type == 6 (VM reset marker), this field = *VM ID*.
Reserved	14	Reserved.
Generation	15	Completion write generation bit. This bit is flipped every wrap-around of the Completion Queue write. Software uses this bit to identify newly-written completions versus those that have already been processed.
Tx Head	28:16	Current Tx-Queue's head. Points to the next pending descriptor. Valid when *Completion Type* is one of 000b–100b.
Completion Type	31:29	Completion entry type (this is a coding of completion types): 000b = Tx Head report as part of ITR flush process (No *RS*, No *RE*). 001b = Tx completion upon *RS* bit set in the Tx-Descriptor. 010b = Tx completion upon *RE* bit set in the Tx-Descriptor. 011b = Tx completion upon *RS* + *RE* bits (both bits are set in Tx-Descriptor). 100b = Drop notification upon transmit timestamp expiration, drop doorbell request or NOP descriptor. ***Note:*** When Drop completion is about to be reported, if one or more packets are successfully transmitted and not yet reported, an unsolicited completion is reported for the prior reporting of the drop. In that case, *Completion Type* 0 (000b) is used for this unsolicited. Other Drop events, like anti-spoof, are not reported as a "Drop" completion. 101b = Queue Disable marker. Marks that Tx-Queue disable flow is flushed out via the Completion Queue. In that case, *RS* and *RE* bits from Tx-Descriptor, are not reflected. 110b = VM reset marker. When a VM reset flow is completed and any activity related to any queue of the VM is drained from the Tx-Pipe, a VM reset marker is posted in any Completion Queue that is associated with this VM. ***Note:*** This marker is triggered by EMP firmware. 111b = Reserved. Must not be used.

Unlike many other descriptors that implement a Descriptor Done (DD) bit, software identifies new Completion Queue entries by comparing the generation bit. Hardware toggles the generation bit on every wrap-around on the Completion Queue. Using this method, software can determine the old entries from the new ones by looking for mis-matches on the generation bit from one descriptor to the next.

Software must configure the Completion Queue long enough to ensure that completion is not overlapped before software processes them.

10.5.4.6 Doorbells

E810 Tx-Queues can be configured in one of two Host interface modes: Legacy and Advanced. While the doorbell registers are shared for both modes, the doorbell format is different in those two modes, Hitting a doorbell to a specific queue is done by CSR writing to its doorbell register.

10.5.4.6.1 Legacy Mode Doorbell

When a queue is used in legacy host interface mode, the doorbell format is like in legacy devices, where writes to a queue's tail register simply reflect advancing of the tail pointer in the transmit ring to post new descriptors to hardware for processing.

Table 10-22. Legacy Doorbell Format

Field	Bit(s)	Init.	Type	Description
TAIL	12:0	0x0	RW	The Transmit Tail defines the first descriptor that the software prepares for the hardware (it is the last valid descriptor plus one). The Tail is a relative descriptor index to the beginning of the transmit descriptor ring.
Reserved	31:13	0x0	RSV	Reserved. Must be set to zero.

10.5.4.6.2 Advanced Mode Doorbell

When a queue is configured for advanced host interface mode, writes to the queue tail register are redefined to pass additional information to control packet scheduling via quanta descriptors. Furthermore, the format of tail writes is different between Single Send (SSO) and Transmit Segmentation Offload (TSO) operations.

In addition, advanced mode supports current head-of-queue dropping based on quanta descriptors of scheduled packets via a Drop Doorbell format described below. An expected case is if the transmit backlog on a given queue grows too long, software can instruct the hardware to ignore existing packets posted on the transmit queue for transmission. Software specifies the number of quantas to drop. Hardware automatically selects the packets to drop from the current head-of-queue. Hardware then generates completion notifications to software to indicate which specific packets were dropped. Hardware accumulates the drop requests and executes the drops when the Tx-Queue is scheduled for transmission.

The internal Drop accumulator is a 7-bit counter per Tx-Queue. It can accumulate up to 127 quantas drop request. Overflow is silently ignored. When the Tx-Queue is empty, any Drop request is ignored and the accumulator is reset. Advanced Mode also includes the ability for software to specify a maximum guaranteed transmission time specified as an expiration time for transmitted packets. If packets are not transmitted in time, the packets are automatically dropped by the transmit pipeline.

The completion notification is identical to the method described above. To enable using this time based transmission, software must maintain a free running timer of 18 bits with resolution of 1 ms. Software is required to read this hardware timer from time to time and synchronize its timer.

When queuing a quanta with *Expire Timestamp*, the timestamp data is part of the quanta descriptor. When queuing it, software must keep the written timestamp to be later than current free running time and not later than 18 seconds than current time.

18-bits timer in 1 ms resolution wraps around approximately every four minutes. The assumption is that packets never wait so long in a Tx-Queue. To cover this rare case as well, it is recommended that software periodically scans the Tx-Queue. When finding a packet that pends a long time in the Tx-Queue, software can send a drop request to that Tx-Queue.

When adding a packet to the Tx-Queue with an expiration time, the delta time between the expiration time of the entered packet and the previous packet in the queue must be smaller than 22 seconds.

Note: Legacy queues or queues with TSO enabled (Mode 1 or 2) do not support "Head Drop". Queues that use drop mode must also enable Completion Queue notifications.

Table 10-23. SSO Doorbell Format

Field	Bit(s)	Init.	Type	Description
TAIL	12:0	0x0	RW	The Transmit Tail defines the first descriptor that the software prepares for the hardware (it is the last valid descriptor plus one). The Tail is a relative descriptor index to the beginning of the transmit descriptor ring.
DB Type	14:13	00b	RW	Doorbell Type 00b = SSO (default) 01b = LSO 10b = Drop doorbell 11b = Reserved. Hardware silently ignores this doorbell.
Reserved	15	0b	RSV	Reserved.
Data Length	23:16	0x0	RW	Data length incorporated in current doorbell in 64-byte units, rounded up. For 65-byte packet, *Data Length* field == 2.
Number of Packets	29:24	0x0		Number of packets incorporated in current doorbell. A quanta can include 1-63 packets. a value of 0 is reserved for special cases.
Quanta Completed	30	0b	RW	Set to 1b if this doorbell closes a full quanta.
Reserved	31	0b	RSV	Reserved.

Table 10-24. TSO Doorbell Format

Field	Bit(s)	Init.	Type	Description
TAIL	12:0	0x0	RW	The Transmit Tail defines the first descriptor that the software prepares for the hardware (it is the last valid descriptor plus one). The Tail is a relative descriptor index to the beginning of the transmit descriptor ring.
DB Type	14:13	00b	RW	Doorbell Type 00b = SSO (default) 01b = LSO 10b = Drop doorbell 11b = Reserved. Hardware silently ignores this doorbell.
Reserved	15	0b	RSV	Reserved. Must be set to 0b.
First Quanta Length	23:16	0x0	RW	Data length that is sent in each quanta during this LSO transmission. This does not include the last quanta (when the TSO spans on more than one quanta), which can be shorter and covers the tail of the LSO message. The quanta size is in 64-byte units and it is rounded up. In Queue Type #2 (LSO enabled), the max quanta size is 2 KB when transmitting TSO packets.
First Quanta Number of Segments	29:24	0x0	RW	Number of segments that are sent in each quanta during this LSO transmission. This does not include the last quanta (when the TSO spans on more than one quanta), which can be shorter and covers the tail of the LSO message.
Reserved	30	0b	RSV	Reserved. Must be set to 0b.
Single Quanta TSO	31	0b	RW	Set when first quanta is also the last quanta, in case a Large Send packet contains a single quanta.

Table 10-25. Drop Doorbell Format

Field	Bit(s)	Init.	Type	Description
Number of Quantas to Drop	6:0	0x0	RW	Drop command work in quanta granularity. This field marks how many quantas to drop. a value of 0 in this field is Illegal and is silently be ignored by hardware. Software can instruct the hardware to drop 1-127 Quanta's. If this queue is configured to Completion Queue method, the Drop Done (DD) is reported by the Completion Queues. If the queue is configured for the other completion methods, the Drop takes effect with no Done reporting. The hardware manages a Drop register per each Tx-Queue. This is a 7-bit register. When this register is saturated, the hardware uses the saturated value and drops the reminder with no overflow notification to software. It is recommended that software checks the Drop status in the Completion Queue before posting more Drop commands.
Reserved	12:7	0x0	RSV	Reserved.
DB Type	14:13	00b	RW	Doorbell Type 00b = SSO (default) 01b = LSO 10b = Drop doorbell 11b = Reserved. Hardware silently ignores this doorbell.
Reserved	31:15	0x0	RSV	Reserved.

Note: Drop commands are supported only for Tx-Queues type #3. Drop doorbell for a Tx-Queue type #1 or #2 are considered as a malicious behavior.

10.5.4.7 Doorbell Queue Descriptors

At very high packet rates, register writes for each packet to a device doorbell register becomes inefficient. A common software practice is to batch tail writes to post several packets at a time to hardware for processing. However, with high queue counts (16K transmit queues), the tail writes themselves can be spread out over a number of queues at one time, making an effective software tail batch algorithm difficult to implement. The E810 supports a mode where the individual doorbell writes, which could be posted directly to the tail register, instead are coalesced into a series of doorbell descriptors stored in a doorbell descriptor ring. From a single queue, software can advance the tail pointers on multiple queues simultaneously, effectively posting multiple packets to various queues with a single Doorbell Queue tail write.

Table 10-26. Doorbell Queue Doorbell Format

Field	Bit(s)	Init.	Type	Description
TAIL	12:0	0x0	RW	The Transmit Tail defines the first descriptor that the software prepares for the hardware (it is the last valid descriptor plus one). The Tail is a relative descriptor index to the beginning of the transmit descriptor ring.
Reserved	31:13	0x0	RSV	Reserved. Must be set to zero.

Table 10-27. Doorbell Descriptor Format

Field	Bit(s)	Init.	Type	Description
TXQ_ID	13:0	0x0	RW	Target Transmit Queue ID.
DD	14	0b	RW	Descriptor Done bit. Set to 0b by software, Set to 1b by hardware
RS	15	0b	RW	Report Status.
DUMMY	16	0b	RW	Used for marker. Treated as a no-op by hardware.

Table 10-27. Doorbell Descriptor Format [continued]

Field	Bit(s)	Init.	Type	Description
Reserved	31:17	0x0	RSV	Reserved.
DBData	63:32	0x0	RW	Doorbell Data. The exact format of a doorbell as it would sent in the Direct Doorbell. All three doorbell formats can be sent via the Doorbell Queue.

A single Doorbell Queue serves one and only one PF or VF. Software advances the Doorbell Queue by writing to the Doorbell Queue Doorbell CSR.

When processing a Doorbell Descriptor, if the *RS* bit is set, a descriptor write-back is required right after it was processed. The hardware writes back the value 0xFFFF_FFFF_FFFF_FFFF to the descriptor location.

10.5.4.8 Transmit During Link Down

In most usage models, when the link becomes inactive, it is not required to preserve the transmit packets that were not sent. During link down, the hardware continues to fetch transmit descriptors and data regardless of the link status. Packets directed to the network are discarded, while packets directed to local VSI port are forwarded successfully.

Note: When the link becomes inactive, PFs on this port get a link status change interrupt. It is expected that the VFs get the link status change indication from their parent PF.

10.5.5 Transmit Configuration

This section covers the various initialization and configuration requirements and examples of the following areas:

- Transmit queue context configuration.
- Quanta descriptor queue context configuration, and associated quanta profile context.
- Doorbell Queue configuration.
- Completion Queue context configuration.

10.5.5.1 Transmit Queue Programming

Queue enable and disable flows are described and summarized in Table 10-31 on page 1342.

Unlike previous products, which used host memory to cache queue contexts, the E810 implements all queue contexts in on-die internal memories. In response to an Add Queue Admin Queue command, EMP firmware uses a series of indirect register accesses (a pair of index and data registers) to program the queue contexts. The flow is described below.

10.5.5.2 Transmit Queue Context

This section describes the setting options of the LAN transmit queue parameters named as "queue context".

Transmit Q context is an on-die structure that contains the configuration, state, and internal scratch pad of each one of the transmit queues.

Context configuration change cannot be performed while the Tx-Queue is active. Configuration is allowed only when the queue is disabled or suspended (in the Tx-Scheduler).

10.5.5.2.1 Transmit Queue Context Structure

Table 10-28. LAN Tx-Queue Context in the QTXCOMM_CNTX Array

Alias	Type	Width (Bits)	LS Bit	MS Bit	Description
Queue_addr	Static	57	0	56	Descriptor queue 57 MS bits of the start address. Every queue address must be 128B-byte aligned. 7 LS bits are 0.
PORT_num	Static	3	57	59	Port number with which the queue is associated.
CGD_num	Static	5	60	64	CGD number with which the queue is associated.
PF_num	Static	3	65	67	PF number that owns this queue.
VMVF_NUM	Static	10	68	77	VF/VM index should be programmed per *VMVF_TYPE* setting: • For *VMVF_TYPE*=VF, it is the VF number between 0:256. • For *VMVF_TYPE*=VM, it is the VM number between 0:767. • For PF or EMP, this field should be set to zero.
VMVF_TYPE	Static	2	78	79	VF/VM Type 00b = Queue belongs to VF. 01b = Queue belongs to VM. 10b = Queue belongs to PF. 11b = Reserved.
SRC_VSI	Static	10	80	89	The VSI with which the queue is associated (0: 767).
TSYN_ENA	Static	1	90	90	TimeSync (1588) enabled via this queue.
Internal Usage Flag	Static	1	91	91	This Flag is reserved for internal usage. This bit must be equal to the *TSO_Enabled_Queue* flag.
ALT_VLAN	Static	1	92	92	Alternate VLAN tag select.
CPUID	Static	8	93	100	CPU ID for TPH. If enabled, it is updated by the read completion of transmit data rather than the descriptors, as implemented in the legacy LAN queues.
WB_MODE	Static	1	101	101	Select between descriptor write-back and Completion Queue write-back: 0b = Descriptor write-back. 1b = Completion Queue write-back.
TPHRDdesc	Static	1	102	102	Descriptor read TPH enable.
TPHRDdata	Static	1	103	103	Data read TPH enable.
TPHWRdesc	Static	1	104	104	Descriptor write TPH enable.
CMPQ_ID	Static	9	105	113	Completion Queue ID.
qnum_in_func	Static	14	114	127	Queue ID in PF or VF space, as reported in Completion Queue.

Table 10-28. LAN Tx-Queue Context in the QTXCOMM_CNTX Array [continued]

Alias	Type	Width (Bits)	LS Bit	MS Bit	Description
ITR Notification_Mode (Int NoExpire Mode)	Static	1	128	128	ITR Notification Mode: 0b = This queue notifies Tx state when descriptor signed with *RS* or *RE*, and also as part of ITR flow. 1b = This queue notifies Tx state only when descriptor signed with *RS* or *RE*.
adjust_profile_id	Static	6	129	134	Profile ID for packet length Adjustment. See Section 8.3.2.4.2.1 for description.
Queue_length	Static	13	135	147	Transmit Queue Length (QLEN). Defines the size of the descriptor queue in descriptors units from eight descriptors (QLEN=0x8) up to 8K-32 descriptors (QLEN=0x1FE0). QLEN restrictions: At smaller queue size than 32 descriptors, the QLEN must be whole number of eight descriptors. At larger size than 32 descriptors, the QLEN must be whole number of 32 descriptors. ***Note:*** If the queue is configured to Mode #2 or #3 (uses quanta descriptors), its length must be equal to or longer than 32 Tx-Descriptors.
Quanta_profile_idx	Static	4	148	151	Quanta Profile Index. Each Tx-Queue must be associated with one of eight quanta profiles.
TSO_Enabled_Queue	Static	1	152	152	Legacy Queue or Advanced Mode Queue. Setting this bit for Queue Type 1 or 2 enables internal usage of descriptor caching, which is required when Legacy interface or TSO are used.
TSO_Qnum	Static	11	153	163	The E810 supports up to 16K Tx-Queues. 2K of them can be TSO-enabled. Each TSO-enabled queue is associated with a TSO internal state structure. This field is the TSO state pointer. This field must be initialized for every TSO-enabled Tx-Queue. TSO structures are equally shared between the PFs (e.g., in an 8-ports configuration, each PF is associated with 256 TSO context structures). Each PF manages the TSO associating inside across the PF and VF queues. When calling the "Add Tx-Queue" AQC, the software provides a Tx-Queue context for each added Tx-Queue. inside the software structure, software provides a zero-based ID *TSO_Qnum* in the PF space. EMP firmware converts it to a physical *TSO_Qnum* and writes it to the queue context.
legacy interface	Static	1	164	164	Legacy or Advanced Host Interface: 0b = Advanced host interface. 1b = Legacy host interface.
drop enable	Static	1	165	165	When this bit is set to one, the Tx-Queue supports Head Drop feature. For those queues, software must issue the correct *Tx-Descriptor* Field with each quanta descriptor. When the queue is not configured for Drop support, the *Tx-Descriptor* field in the quanta descriptor is ignored.
cache_profile_idx	Static	2	166	167	Association of the queue to a quanta cache profile. Cache profile declare parameter can limit the number of quanta descriptors accumulated in the device cache per the Tx-Queue. This allows a per-queue fine tuning for Drop command response and single queue performance of the queue.

Table 10-28. LAN Tx-Queue Context in the QTXCOMM_CNTX Array [continued]

Alias	Type	Width (Bits)	LS Bit	MS Bit	Description
packet shaper profile idx	Static	3	168	170	Used in Legacy Host interface Queue (Queue Type #1). Some of the Tx-Scheduler nodes are configured for packet shaping. In Legacy queues, the number of packet in the quanta are not known in advance, The packet shaper profile defines a preliminary packets number. The Tx-Scheduler takes this number into consideration until the descriptors' fetch and update flow.
Internal Queue state	Dynamic	122	171	292	Internal queue state.

10.5.5.3 Quanta Profile

The E810 supports 16 quanta profile registers. Every enabled transmit queue is configured with an index to one of the 16 quanta profiles. Each quanta profile specifies resource limits each transmit queue can consume of the shared transmit pipeline. Resource limits include limiting the number of descriptors, commands, and size in bytes allowed per quanta. These limits enable the arbiter and scheduler to guarantee the desired transmit performance of high-priority queues by limiting sources of disruption from lower-priority queues.

The quanta Profiles span on 16 registers: GLCOMM_QUANTA_PROF[0..15]. The quanta profiles are auto-loaded from the NVM. They can be reconfigured by the active PFs. When reconfigured by a PF, the ownership is managed by the PFs itself.

The Field *QUANTA_SIZE* in the Profile is used to configure the quanta size in bytes. Used by hardware for LSO processing and for legacy queue. When it configures a legacy queue quanta size, the value must configured in 64-byte granularity (The five LSB must be set to 0).

The *QUANTA_SIZE* field is configured to a value between 256 bytes and 4 KB.

Notes:

- GLLAN_TCLAN_CACHE_CTL.*MIN_ALLOC_THRESH* represents descriptors fetching caching policy. This register needs to be configured to (the MAX between all Number of descriptors in a quanta from all Quanta Profiles) + 4.
- It is recommended that legacy queues will be configured to 1 KB quanta.

10.5.5.4 Quanta Descriptor Cache Profile

Cache profiles are used to fine tune quanta descriptor caching between highest single queue performance and faster response to Head Drop.

There are four quanta descriptor cache profiles used to configure the quanta descriptor cache behavior of any given transmit queue. As the case with most other descriptors, posted quanta descriptors are read ahead into local memory for fast access and processing by the scheduler. The quanta descriptor cache itself is a limited and shared resource. Cache profiles enable software to bias device processing preference to high priority queues over best effort queues.

Quanta descriptor caching has two related potential impacts to device performance. If insufficient cache is allocated to a given cache profile, single-queue transmit performance can suffer from excessive quanta descriptor fetch latency. Likewise, any queue that is subject to drop quanta doorbell commands from software should point to a cache profile that limits the number of pre-fetched quanta descriptors. Drop quanta doorbell commands can introduce unnecessary bubbles in the transmit pipeline as pre-fetched quanta must be drained from the transmit pipeline.

The Cache Profiles span on four registers: GLQDC_CFG_PRFL_1[0..3]. The cache profiles are auto-loaded from the NVM. They can be reconfigured by the active PFs. When reconfigured by a PF, the ownership is managed by the PF itself.

10.5.5.5 Packet Shaping Profile

Packet Shaping profile is used for Legacy Queues. When packet shaping is used, when queue is configured with advanced mode host interface, the number of packets in a quanta is provided by software in the doorbell and in the quanta descriptor.

If a queue is configured to legacy host interface, the number of packets is not provided by software. The queue is associated with Packet Shaping Profile, which provides a prediction per usage model queue type of the number of packets in a quanta, where its size is configured in Quanta Profile.

Packet Shaping Profile is an 8-CSRs structure: GLCOMM_PKT_SHAPER_PROF[0..7].

10.5.5.6 Doorbell Queue Configuration

Software is required to fully initialize the entire Doorbell Queue Context to zero prior to enabling the Doorbell Queue. The Doorbell Queue configuration can only be changed while the associated transmit queues are disabled. Read/Write operations to the context structure are performed via direct register access to the Doorbell Queue Configuration context QTX_COMM_DBLQ_CNTX[5][256].

The Doorbell Queue Context structure is detailed in Table 10-29.

Table 10-29. Doorbell Queue Context

Alias	Width (Bits)	LS Bit	MS Bit	Type	SW Init.	Description
Queue_addr	57	0	56	Static	Variable	Descriptor queue start address.
Reserved	7	57	63			Reserved.
Queue_length	13	64	76	Static	Variable	Queue length in descriptor units.
Reserved	3	77	79			Reserved.
PF_NUM	3	80	82			PF number.
Reserved	1	83	83			Reserved.
VF_NUM	8	84	91	Static	Variable	VF number.
Reserved	2	92	93			Reserved.
PFVF_TYPE	2	94	95			00b = Queue belongs to VF. 01b = Reserved. 10b = Queue belongs to PF. 11b = Reserved.
CPUID	8	96	103	Dynamic		
TPH Desc Rd	1	104	104	Static		
Reserved	3	105	107			Reserved.
TPH Desc Wr	1	108	108	Static		
Reserved	3	109	111	Static		Reserved.
DBL_Q_ENA	1	112	112	Static		Doorbell Queue Enable. When disabled, the queue is not scanned by the device.
Reserved	15	113	127			Reserved.

Table 10-29. Doorbell Queue Context [continued]

Alias	Width (Bits)	LS Bit	MS Bit	Type	SW Init.	Description
Read Head	13	128	140	Dynamic		Queue Head. Read by software. Written by hardware.
Reserved	3	141	143			Reserved.
Read Tail	13	144	156	Dynamic		Queue Tail. Written by software. Read by hardware.
Reserved	3	157	159			Reserved.

10.5.5.7 Completion Queue Configuration

Unlike many other queue usage models, Completion Queues are expected to be associated with a given CPU core. The expected usage model assumes that software is bound to a given CPU core, and furthermore schedules data for transmission on one or more transmit queues. The Completion Queue serves to aggregate transmit completions from multiple transmit queues to a single Completion Queue.

The Completion Queue is associated with one and only one PF or VF. All transmit queues that can post completion events must be associated with this associated PF or VF. The completions can be generated at various rates depending on the per-queue ITR settings.

Software is required to fully initialize the entire Completion Queue context structure to zero prior to enabling of the Completion Queue. The queue configuration can only be changed after all associated transmit queues are disabled. Read/Write operations to the context structure are performed via direct register access to the Completion Queue Configuration context GLTCLAN_CQ_CNTX[21][512].

The Completion Queue context structure is detailed in Table 10-30.

Table 10-30. Completion Queue Context

Field	Bit(s)	Type	Description
Queue Address	56:0	RW	Base Address of the Completion Queue. Completion Queue base must be cache-line aligned. This field does not contain the low seven bits of the address, which are always zero.
Reserved	63:57	RSV	Reserved.
Queue Length	81:64	RW	Completion Queue length in units of 16 completion descriptors. The Completion Queue length is a multiple of catch-lines. This field does not contain the low four bits of real Completion Queue length. Max Completion Queue length is 4M entries. This high number is to cover completion from multiple transmit queues in case software is delayed in completion entries processing.
Reserved	95:82	RSV	Reserved.
Generation	96	RW	Current generation bit. This bit is flipped every wraparound event.
WritePTR	118:97	RW	Current completion write pointer (0. 4M-1).
Reserved	127:119	RSV	Reserved.
PF_NUM	130:128	RW	PF number.
VMVF_NUM	140:131	RW	VM/VF number.
VMVF_TYPE	142:141	RW	00b = Queue belongs to VF. 01b = Queue belongs to VM. 10b = Queue belongs to PF. 11b = Reserved.

Table 10-30. Completion Queue Context [continued]

Field	Bit(s)	Type	Description
Reserved	159:143	RSV	Reserved.
TPH Desc Wr	160	RW	
CPUID	168:161	RW	
Reserved	191:169	RSV	Reserved.
Internal completions cache	703:192	RW	Internal cache of completions. Completions are written to the host as a full, aligned cache-line. If a write of a partial cache-line is required, the remaining bytes are padded out with zeros and with the opposite generation bit, such that software does not consider the padding as a completion.

Some other commands can be operated for a Completion Queue, like initiating VM reset marker and flushing out of internal Completion Queue cache. Those commands are operated using CSRs GLCOMM_CQ_CTL[512].

10.5.5.8 Tx-Queue Handling Admin Queue Commands

Table 10-31. Tx-Queue Handling Admin Queue Commands

Command	Opcode	Brief Description	Section Reference
Add Tx-Queues	0x0C30	This command configures queues in the PQM, configures leaf nodes in the PSM, and associates the queues with the leaf nodes. This command allocates leaf nodes. Software provides the parent node's TEID and number of queue to allocate. With the completion, a separate TEID is provided per each added queue.	10.5.5.8.1
Queues Disable	0x0C31	This command gets a list of QIDs, disables the Tx-Queues, and releases relevant Tx-Scheduler resources. This command serves both LAN queues and RDMA QSets	10.5.5.8.3
Move/Reconfigure Tx-Queues	0x0C32	This command reconfigures Tx-Queues in the E810. This AQ can be used to: 1. Restructure the Tx-Scheduler and move a leaf node or a group of leaf node from on parent to another. 2. Reconfigure the Tx-Queues' CGDs settings. 3. Both 1 and 2.	10.5.5.8.4
Add RDMA Queue Set	0x0C33	This command configures leaf nodes in the PSM and associate the queue sets with the leaf nodes. This command allocates leaf nodes. software provides the parent node's TEID and number of queue sets to allocate. With the completion, a separate TEID is provided per each added queue set.	10.5.5.8.2
Move RDMA Tx-Queue Sets	N/A	***Note:*** This is covered by Tx-Scheduler AQ command "Move Node". RDMA QSet CGD is managed inside the PE.	

10.5.5.8.1 Add Tx LAN Queues (0x0C30)

This command configures Tx-Queues in the E810, configures leaf nodes in the PSM, and associates the queues with the leaf nodes.

The command gets the following as input parameters:

- Number of added Queues Groups (a Queue Group is a group of queues that are added to a single Scheduling Parent).
- List of Parent Nodes' TEIDs. Each is the handle of the parent node of all added queues.
- Number of added queues in each Queue Group.
- List of Queue IDs (in the PF space).
- A queue context structure for each one of the queues.
- Tx-Scheduler leaf node bandwidth configuration parameters for each leaf node associated with add queue.

This command allocates leaf nodes. Software provides the parent node's TEID and number of queues to allocate. Upon completion, a separate TEID is provided per each added queue.

This is an Indirect Admin Queue command, with additional command attributes and completion attributes provided within the data buffer. Table 10-32 describes command format and defines command specific fields.

Table 10-32. Add Tx LAN Queue Command Fields

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0C30	
Datalen	4-5		Length of response buffer.
Return Value	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Number of Queue Groups	16		Number of added queue groups. Each added queue group is associated with a structure inside the command buffer containing the parent TEID, the number of added queues in the group, and their queue IDs A maximum of 73 queue groups can be added in one AQ call. Since the command buffer containing the Queue Groups Structures is bounded to 4 KB, when adding many queues in each queue group, this number can be limited to the lower number of queue groups. 0 is an invalid value for this field. In the command response, this field returns the number of successfully-added queue groups (the first N added queue groups).
Reserved	17-23	0	Reserved. Must be set to 0.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Table 10-33 describes format of the data buffer carrying additional command attributes and the response buffer.

Table 10-33. Add Tx LAN Queue Command and Response Buffer

Category	Byte.Bit	Field	Description
Per Queue Group Structure			Per each added queue group, a structure.
	0-3	Parent's TEID	The TEID of the parent node to which leaves are added.
	4	Number of Queues	Number of added queues (1..85).
	5-7	Reserved	Reserved.
Per-Queue Structure			Per-queue 48-byte structure.
	0-1 in per-queue structure	Tx Q ID	Tx-Queue ID in LAN space. The number is in PF space. Firmware is required to translate to global numbering.
	2-3	Reserved	Reserved. Must be set to 0.
	4-7	Queue TEID	In the command: Reserved In the response: Added Tx-Queue TEID.
	8-29	Tx-Queues Context Structure	Per each added queue, raw data structure of Q context 170 bits. The "Internal Queue State" is not included. It is reset by hardware.
	30-31	Reserved	Reserved. Must be set to 0.
Tx-Scheduler Leaf Node Configuration			The flow of adding a Tx-Queue includes adding of a Tx-Scheduler leaf node associated with it. The structure below is this node's configuration. The structure is identical to the node's configuration structure in "Add Scheduling Component" AQC.
	32	Reserved	Reserved. Must be set to 0.
	33	Valid Sections	Multiple sections can be valid at a given time. **Bit(s) Description** 0 = Generic section (Must be set to 1b.) 1 = CIR BW 2 = EIR BW 3 = Shared BW 4-7 = Reserved. When only the *EIR BW* (bit 2) is set, firmware configures EIR bandwidth only. When only the *Shared BW* (bit 3) is set, firmware configures SRL ID and switch node to SRL. When both *EIR BW* and *Shared BW* are set, firmware configures EIR, SRL and switch to SRL.
	34	Generic	**Bit(s) Description** 0 = Scheduling mode BPS (0) or PPS (1). 1-3 = Priority among siblings (0-7). 4 = Single priority node (1) or WFQ (0). 5-6 = Adjustment value (0-3) used in PSM Credit Update flow. 7 = Reserved.
	35	Reserved	Reserved.
	36-37	CIR BW Profile ID	
	38-39	CIR BW WFQ Weights (1-200)	
	40-41	EIR BW Profile ID	When the valid section has *Shared BW* set, this field is treated as reserved.
	42-43	EIR BW WFQ Weight (1 -200)	
	44-45	Shared RL Profile ID	When the valid section has *EIR BW* set, this field is treated as reserved.
	46-47	Reserved	Padding.

10.5.5.8.1.1 Software Activities Prior to Calling Add Tx LAN Queues AQ

- If a queue uses Completion Queue, then:
  - The Completion Queue ID is part of Tx-Queue context.
  - Completion Queue must be initialized by software and ready prior to associating a Tx-Queue to it.
  - Configuring of a Completion Queue is implemented directly by software with no firmware involvement.
- Quanta profile, Cache profile, and Packet Shaper profile that the Tx-Queue points to must be initialized before the association. This is done by software.
  - Profile configuration is implemented directly by software with no firmware involvement.

10.5.5.8.1.2 Software Activities After Add Tx LAN Queues AQ Completed

- Interrupt association with a specific queue is detailed in Section 9.1.2.1.
- Association of a Tx-Queue to DB queue is not reflected in hardware and not part of Add Tx LAN queue flow.

10.5.5.8.2 Add Tx RDMA Queue Sets (0x0C33)

This command configures leaf nodes in the PSM and associates the queue sets with the leaf nodes.

The command gets the following parameters as input:

- Number of added Queue Sets Groups.
  - Queue Set Group is a group of Queue Sets that are added to a single Scheduling Parent.
- List of Parent Nodes' TEIDs. Each is the handle of the parent node of all added queue sets.
- Number of added Queue Sets in each Queue Group.
- List of Queue Set IDs (in the PF space).

This command allocates leaf nodes. Software provides the parent node's TEID and number of queues to allocate. With the completion, a separate TEID is provided per each added queue set.

This is an Indirect Admin Queue command, with additional command attributes and completion attributes provided within the data buffer. Table 10-34 describes command format and defines command specific fields.

Table 10-34. Add Tx RDMA Queue Sets Command Fields

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0C33	
Datalen	4-5		Length of response buffer.
Return Value	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.

Table 10-34. Add Tx RDMA Queue Sets Command Fields [continued]

Name	Byte.Bit	Value	Remarks
Number of Queue Set Groups	16		Number of added queue set groups. Each added queue set group is associated with a structure inside the command buffer containing the parent TEID, number of added queue sets in the group, and their queue IDs. Maximum 127 queue set groups can be added in one AQ call. The E810 contains maximum 512 RDMA QSets. 0 is an invalid value for this field. In the command response, this field returns the number of successfully-added queue set groups (the first N added queue groups).
Reserved	17-23	0	Reserved. Must be set to 0.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Table 10-35 describes format of the data buffer carrying additional command attributes and the response buffer.

Table 10-35. Add Tx RDMA Queue Sets Command and Response Buffer

Category	Byte.Bit	Field	Description
Per Queue Group Structure			Per each added queue group, a structure.
	0-3	Parent's TEID	The TEID of the parent node to which leaves are added.
	4-5	Number of QSets	Number of added queue sets (1..170).
	6-7	Reserved	Padding.
Per-Queue Structure			Per-QSet 4-byte structure.
	0-1 in per-queue structure	Tx QSet ID	Tx QSet ID in RDMA space. The number is in physical space.
	2-3	Reserved	Reserved.
	4-7	Queue TEID	In the command: Reserved In the response: Added Tx-Queue TEID.
Tx-Scheduler Leaf Node Configuration			The flow of adding a Tx-Queue includes adding of a Tx-Scheduler leaf node associated with it. The structure below is this node's configuration. The structure is identical to the node's configuration structure in "Add Scheduling Component" AQC.

Table 10-35. Add Tx RDMA Queue Sets Command and Response Buffer [continued]

Category	Byte.Bit	Field	Description
	8	Reserved	Reserved. Must be set to 0.
	9	Valid Sections	Multiple sections can be valid at a given time. **Bit(s) Description** 0 = Generic section (Must be set to 1b.) 1 = CIR BW 2 = EIR BW 3 = Shared BW 4-7 = Reserved. When only the *EIR BW* (bit 2) is set, firmware configures EIR bandwidth only. When only the *Shared BW* (bit 3) is set, firmware configures SRL ID and switch node to SRL. When both *EIR BW* and *Shared BW* are set, firmware configures EIR, SRL and switch to SRL.
	10	Generic	**Bit(s) Description** 0 = Scheduling mode BPS (0) or PPS (1). 1-3 = Priority among siblings (0-7). 4 = Single priority node (1) or WFQ (0). 5-6 = Adjustment value (0-3) used in PSM Credit Update flow. 7 = Reserved.
	11	Reserved	Reserved.
	12-13	CIR BW Profile ID	
	14-15	CIR BW WFQ Weights (1-200)	
	16-17	EIR BW Profile ID	
	18-19	EIR BW WFQ Weight (1 -200)	
	20-21	Shared RL Profile ID	
	22-23	Reserved	Padding.

10.5.5.8.2.1 Software Activities Prior to Calling Add Tx RDMA Queue Sets AQ

No specific requirements from software.

10.5.5.8.2.2 Software Activities After Add Tx RDMA Queue Sets AQ Completed

- Interrupt association with a specific Q is detailed in Section 9.1.2.3.
- With the Initialized QSet ID, software uses the QSet ID when calling the related RDMA AQs to associated RDMA QPs with the Initialized QSet.

10.5.5.8.3 Transmit Queue Disable Flow - LAN and RDMA (0x0C31)

In this command, software provides a list of Tx-Queue or RDMA QSet IDs that need to be closed. The AQ must be called after software stops feeding the closed queues (stop sending doorbells). RDMA QSet can be closed only when it is not associated with any QP.

The command gets the following parameters as input:

- Number of disabled Queues/QSets.
- List of Queue or QSet IDs (in the PF space).

This command deallocates nodes (in leaf layer and possibly in intermediate layers as well) in the Tx-Scheduler. Inside the command structure, the disabled queues are organized in groups. Each group includes queues belonging to one parent node in the Tx-Scheduler structure. This organization in group is added to ease the interface between the software and firmware. It is not required to disable all the queues belonging to a parent in one call.

This command is called by software as part of the queue disabling flow. As part of this command flow, the EMP firmware drains the Tx-Pipe of any in-flight packets (packets that are scheduled for transmission by the Tx-Scheduler but have not yet been transmitted) of all disabled Tx-Queues or RDMA QSets. This is required to verify that no Tx completion is posted to software after the queue resources are released or even re-used.

Normally, the pipe draining flow requires a very short delay. A long (or endless) flow control event that blocks the transmit of one or more TCs (or even the entire port transmit) affects the pipe draining flow as well.

This command is completed when all in-flight packets belonging to the Disabled LAN Queues or RDMA QSets have left the Tx-Pipe. Alternately, the command is called with a timeout parameter. If the EMP firmware waits more than the timeout time, it responds with EAGAIN error code. With the EAGAIN error code, firmware also provides a bitmap that marks which of the Congestion Domains is blocked by the long flow control.

Software must re-call this command (with the *Call-Again* flag set) prior to releasing and reusing of any of the Tx-Queues that are associated with the blocked congestion domains.

This command must be fully completed (all Tx-Queue or RMDA QSets are released) prior to any other calling to it for other queues, and prior to calling to “Move/Reconfigure Tx LAN Queues” AQC.

The command’s timeout is posted by software as part of the command’s parameters.

Software can also instruct the EMP firmware to force the Tx-Pipe to flush out and drop all packets from the blocked congestion domain. It is the PF’s responsibility and authority to make the decision when to consider a long flow control as a malicious link partner behavior.

This is an Indirect Admin Queue command, with additional command attributes and completion attributes provided within the data buffer. Table 10-36 describes command format and defines command-specific fields.

Table 10-36. LAN Transmit Queue Disable Command and Response Fields

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0C31	
Datalen	4-5		Length of response buffer.
Return Value	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.
Command Type and Flags	16		**Bits 0:1:** 00b = Queue Disable with no function reset. Must be set to 1. 01b = Reserved. Must be set to 0. 10b = Reserved. Must be set to 0. 11b = Reserved. Must be set to 0. **Bit 2:** 0b = This is an initial call. 1b = This is a subsequent call. **Bit 3:** 0b = Return EAGAIN on timeout. 1b = Flush pipe on timeout. **Bits 4:7:** Reserved. Must set to zero. ***Note:*** Bits 2 and 3 are **NOT** mutually exclusive.
Number of Queue Groups	17		In the command: Number of Disabled Queue Groups (0..127) involved in the Q disable flow. ***Note:*** For the VF/VM reset flow, there might be 0 groups for VFs without queues. In the response: Number of fully-processed groups.
Reserved	18-19.1		Reserved. Must be set to 0.
Timeout Time	19.2-19.7		Command timeout in units of 100 micro seconds. Valid values are 0-50. If this field is 0, the AQ will return “EAGAIN”.
Blocked CGDs	20	0	Zeroed by the driver. A bitmap of blocked CGDs. Set by EMP firmware when returns with EAGAIN.
Reserved	21-23	0	Reserved.
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Table 10-37 describes format of the data buffer carrying additional command attributes. The command buffer includes a structure per involved Queues group.

Table 10-37. LAN Transmit Queue Disable Command Buffer[1]

Category	Byte.Bit	Field	Description
Group #1	0-3	Parent's TEID	The TEID of the parent node to which leaves are involved.
	4	Number of Queues	(1...128).
	5	Reserved	Reserved.
	6-7.6	Queue #1	Tx-Queue ID in PF space of the first queue to be closed.
	7.7	QSet/LAN Queue	Marks if this ID points to a LAN Queue or to a RDMA QSet. 0b = LAN Queue. 1b = RDMA QSet.
	8-9.6	Queue #2	Tx-Queue ID in PF space of the second queue to be closed.
	9.7	QSet/LAN Queue	Marks if this ID points to a LAN Queue or to a RDMA QSet. 0b = LAN Queue. 1b = RDMA QSet.
	. . .		
		Queue #N	Tx-Queue ID in PF space of queue N to be closed.
		QSet/LAN Queue	Marks if this ID points to a LAN Queue or to a RDMA QSet. 0b = LAN Queue. 1b = RDMA QSet.
		Padding	Alignment to 4-byte units.
. . .			
Group #N	0-3	Parent's TEID	The TEID of the parent node to which leaves are involved.
	4	Number of Queues	(1...128).
	5	Reserved	Reserved.
	6-7.6	Queue #1	Tx-Queue ID in PF space of the first queue to be closed.
	7.7	QSet/LAN Queue	Marks if this ID points to a LAN Queue or to a RDMA QSet. 0b = LAN Queue. 1b = RDMA QSet.
	8-9.6	Queue #2	Tx-Queue ID in PF space of the second queue to be closed.
	9.7	QSet/LAN Queue	Marks if this ID points to a LAN Queue or to a RDMA QSet. 0b = LAN Queue. 1b = RDMA QSet.
	. . .		
		Queue #N	Tx-Queue ID in PF space of queue N to be closed.
		QSet/LAN Queue	Marks if this ID points to a LAN Queue or to a RDMA QSet. 0b = LAN Queue. 1b = RDMA QSet.

1. The command buffer include a structure per involved Queues group.

10.5.5.8.3.1 Software Activities Prior to Calling Transmit Queue Disable

- Software must stop sending doorbells to the disabled queues before closing them.
- For Tx-Queues that are associated with Doorbell Queue, the Doorbell Queue must be drained from any doorbell massage 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 with the queue.
  a. When interrupt arrives, software can continue.
- In case of a VM reset or multiple queues disable, software does Step 1 for all disabled queues, and Step 3 for all vectors associated with the queues.

10.5.5.8.3.2 Software Activities After Transmit Queue Disable AQ Completed

- Once the whole operation is complete and the pipe is cleaned from the disabled queues, EMP firmware responds with "No Error" and software can re-use all involved resources.
- Software can post a VM reset notification via the Completion Queue. This is done directly to each involved Completion Queue using CSRs GLCOMM_CQ_CTL[512].

10.5.5.8.4 Move/Reconfigure Tx LAN Queues (0x0C32)

This command reconfigures Tx-Queues in the E810. This AQ can be used to:

1. Restructure the Tx-Scheduler and move a leaf node or a group of leaf nodes from on parent to another.
2. Reconfigure the Tx-Queues' CGDs settings.
3. Both 1 and 2.

The command gets the following parameters as input:

- Number of involved queues.
- Source parent node's TEID.
- Destination parent node's TEID.
- List of queue IDs (in the PF space) and their new CGDs configuration.

By this command, software can instruct the EMP firmware to move nodes belonging to one sibling group. The moved nodes can be the entire group or part of it.

This command is usually called as part of DCBX flow. It allows moving some of the queues between TCs or CGDs.

As part of this flow, the Tx-Scheduler structure is updated in parallel to changing the CGD setting of the LAN Tx-Queues. Moved RDMA QSet CGD setting is managed internally in the RDMA block. It is under software responsibility to call the RDMA CQP for the updating of the moved RDMA QSet contexts.

When CGD association of LAN queue is changed as part of this command flow, the EMP firmware drains the Tx-Pipe of any in-flight packets (packets that are scheduled for transmission by the Tx-Scheduler but have not yet been transmitted) of all updated Tx-Queues. This is required to prevent out-of-order transmission or completion.

Normally, the pipe draining flow requires a very short delay. A long (or endless) flow control event that blocks the transmit of one or more TCs (or even the entire port transmit) affects the pipe draining flow as well.

This command is completed when all in-flight packets belong to the disabled LAN queues or RDMA QSets left in the Tx-Pipe. Alternately, the command is called with a timeout parameter. if the EMP firmware waits more than the timeout time, it responds with EAGAIN error code. With the EAGAIN error code, firmware also provides a bitmap that marks which of the Congestion Domains is blocked by the long flow control.

Software must re-call this command (with the *Call-Again* flag set) prior to updating any of the Tx-Queues that are associated with the blocked congestion domains.

This command must be fully completed (all Tx-Queues drained) prior to any other calling to it for other queues, and prior to calling to "Transmit Queue Disable Flow (LAN and RDMA)" AQC.

The command's timeout is posted by software as part of the command's parameters.

Software can also instruct the EMP firmware to force the Tx-Pipe to flush out and drop all packets from the blocked congestion domain. It is the PF's responsibility and authority to make the decision when to consider a long flow control as a malicious link partner behavior.

Note: This command is an atomic command. EMP firmware must verify it can complete the entire command prior updating any Tx-Queue context or moving any element in the Tx-Scheduler hierarchy.

This is an Indirect Admin Queue command, with additional command attributes and completion attributes provided within the data buffer. Table 10-38 describes command format and defines command specific fields.

Table 10-38. Move/Reconfigure Tx-Queues Command and Response Fields

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 9.5.5.2.1 for details.
Opcode	2-3	0x0C32	
Datalen	4-5		Length of response buffer.
Return Value	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the firmware into the completion of this command.

Table 10-38. Move/Reconfigure Tx-Queues Command and Response Fields [continued]

Name	Byte.Bit	Value	Remarks
Command Type and Flags	16		**Bits 0:1:** 00b = Reserved. Must not be used. 01b = Move nodes with no TC change. 10b = TC change, with no nodes movement. 11b = Both nodes movement and TC change. **Bit 2:** 0b = This is and initial call. 1b = This is a subsequent call. **Bit 3:** 0b = Return EAGAIN on timeout. 1b = Flush pipe on timeout. **Bits 4:7:** Reserved. Must set to zero. ***Note:*** Bits 2 and 3 are **NOT** mutually exclusive.
Number of Queues	17		Number of moved queues. In the command response, this field returns the number of successfully-moved queues (the first N moved queues).
Reserved	18-19.1		Reserved. Must be set to zero.
Timeout Time	19.2-19.7		Command timeout in units of 100 micro seconds. Valid values are 0-50. The two LSB are unused to keep coherency with other commands.
Blocked CGDs	20	0	Zeroed by the driver. A bitmap of blocked CGDs. Set by EMP firmware when returns with EAGAIN.
Reserved	21-23	0	Reserved
Data Address High	24-27		Address of buffer.
Data Address Low	28-31		

Table 10-39 describes format of the data buffer carrying additional command attributes and the response buffer

Table 10-39. Move/Reconfigure Tx-Queues Command and Response Buffer

Category	Byte.Bit	Field	Description
	0-3	Source TEID	Source Parent TEID. The parent of the leaf nodes before moving.
	4-7	Destination TEID	Destination Parent TEID. The parent of the leaf nodes after moving.
Per-Queue Structure			Per-queue 8-byte structure.
	0-1 in per-queue structure	Tx Q ID	Tx Q ID in LAN. The number is in PF space. Firmware is required to translate to global numbering
	2	Queue CDG	0..7
	3	Reserved	Reserved. Padding. Must be set to zero.
	4-7	Queue TEID	Tx-Queue Leaf node TEID.

10.5.5.8.4.1 Software Activities Prior to Calling Move/Reconfigure Tx LAN Queues AQ

No specific requirements from software.

10.5.5.8.4.2 Software Activities After Move/Reconfigure Tx LAN Queues AQ Completed

No specific requirements from software.

10.5.6 Packet Transmission

During nominal operation, the function that owns the queue (PF or VF) accesses the hardware directly. To transmit a packet, software prepares transmit descriptors starting at the descriptor indicated by the *TQTAIL* pointer in the relevant QTX_TAIL register. The descriptors point to the transmitted packet. After the descriptors are ready in the Tx-Queue, software notifies the hardware. This notification is called a doorbell. The doorbell flow is different between the legacy mode and the advanced mode, as detailed in Section 10.5.6.1 and Section 10.5.6.2. A doorbell can be pushed directly to the proper address in the PCIe space or to issue it into the Doorbell Queue, coalescing multiple doorbell notifications in the Doorbell Queue before issuing a Doorbell Queue doorbell to push them to the device.

10.5.6.1 Transmit Doorbell Flow in Legacy Mode

1. Software updates queue tail (*TQTAIL*) using registers QTX_COMM_DBELL [16K].
2. Software updates *TQTAIL* at whole structures boundaries. For TSO, it is the whole TSO; for a single packet, it is the whole packet; and for filter programming, it is the whole packet that is associated with the filter programming. Updating *TQTAIL* to point into the middle of a multi-descriptor operation can trigger a Malicious Driver Detection (MDD) event and halt the queue.
3. The software should never set the *TQTAIL* to a value above the descriptors owned by the hardware minus 1. Descriptors considered as "owned by the hardware" are those already indicated to the hardware, but are not yet reported as completed. Overrunning the queue triggers an MDD event.
4. The hardware reports completed descriptors only for those ones indicated by an *RS* or *RE* (or both) bit in the *CMD* field in the descriptor. As part of the ITR expire flow, hardware reports the latest head, even if it is not marked with *RS* or *RE* bits. Completion indication is provided in one of two modes as programmed by the *HEAD_WBEN* parameter in the queue context as follows:
   - Descriptor Write-Back: The hardware changes the value of the *DTYPE* field in the completed descriptor to a 0xF (*DTYPE* equals to 0xF is reserved for completed descriptor indication). Besides the *DTYPE* field, the rest of the descriptor fields can remain as is or be changed by hardware.
   - Completion Queue Event: For details of the data written back, see Section 10.5.4.5, "Completion Queue (CQ) Descriptor".

10.5.6.2 Advanced Transmit Mode

Advanced transmission is a per Tx-Queue configured mode. In advanced mode, the actual quanta request is provided by software prior to scheduling. This enables the Tx-Scheduler to make more accurate decisions and to more efficiently control this queue's burstiness.

As described in Section 10.5.4.4, quanta descriptors describing the size of a quanta are queued to the device prior to en-queuing data packets for transmission (doorbell). Software uses two interfaces to feed the scheduled quanta to the device.

- **Doorbell** — For each new packet or packets, the software writes to a CSR QTX_COMM_DBELL[16K], advancing the transmit descriptor ring tail pointer and providing additional metadata used to schedule packets.
- **Quanta Descriptor** — Detailed in Section 10.5.4.4.

The following subsections detail the Doorbell and the Quanta Descriptor.

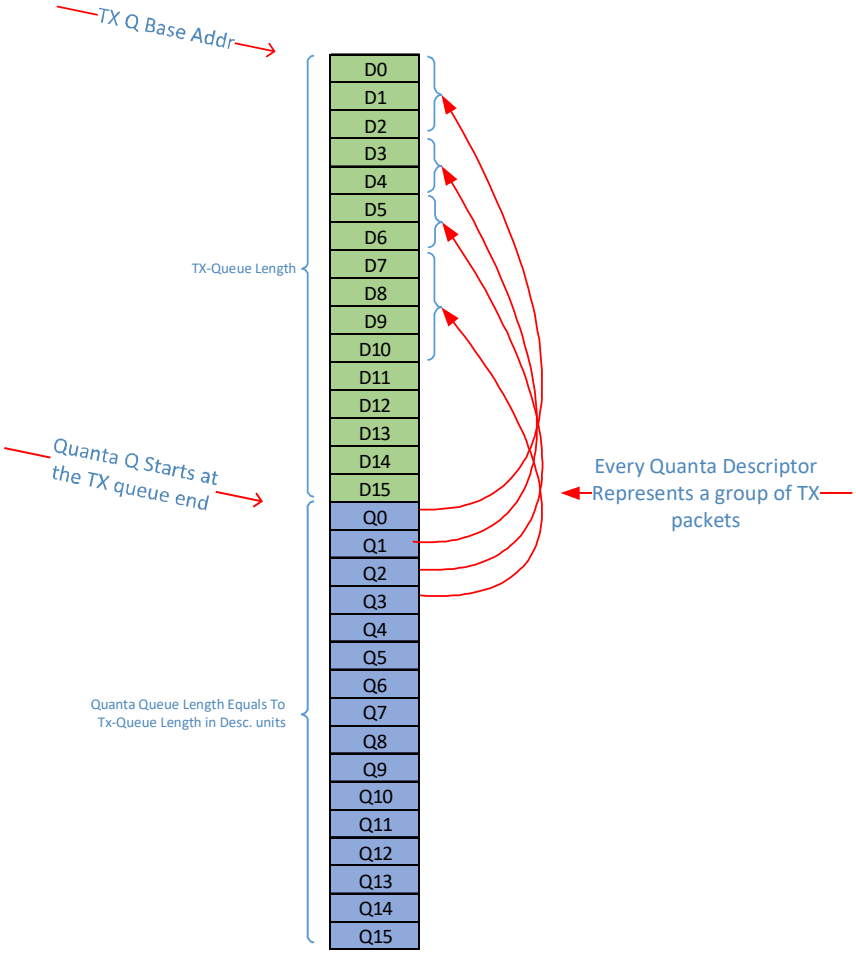


Figure 10-10. Quanta Descriptor and Data Descriptor

Note: When the transmit descriptor queue is smaller than 32, software must complete and close a quanta for every transmitted packet.

Prior to transmitting data on a transmit queue, software is assumed to have a queue-specific burst size, which is defined as the quanta size. The quanta size, in addition to the quanta profile, is used as a hint by the device to implement the WFQ transmit arbitration and scheduling algorithm. In addition, an associated Quanta Descriptor Ring (QDR) is initialized by software. The quanta queue reflects the actual transmit data posted by software to a given transmit queue.

Software allocates a Transmit Descriptor Ring (TDR), followed immediately and contiguously by the corresponding QDR. Unlike the TDR, the QDR does not have explicit head and tail pointer registers. Instead, software implicitly advances the QDR tail pointer by completing one quanta context in the TDR doorbell write. Every transmit descriptor queue, when operated in advanced transmit mode, must have a corresponding quanta descriptor ring of equal length.

As software initializes transmit descriptors in the transmit ring, software writes to the associated transmit queue doorbell register, posting one or multiple transmit descriptors. In addition, software initializes a quanta descriptor in the QDR to track the data bytes consumed per quanta.

The quanta size can change from one quanta to the next. As additional packet headers can be transparently added to transmitted packets by hardware, hardware also tracks the actual physical bandwidth used on a per packet basis when managing the posted quanta descriptors.

For SSO packets, software posts a quanta descriptor under these conditions:

- Whenever the total amount of pending transmit data equals or exceeds the expected quanta size.
- Whenever the total number of posted transmits is about to exceed 54 descriptors.
- Whenever a TCP Segmentation Offload is requested, the E810 automatically closes a previous quanta context and starts a new quanta descriptor context.

Under the expected usage model for this operating mode, software is expected to generally batch doorbell writes to the device spanning multiple transmit descriptors, and close out quanta descriptors simultaneously.

For non-TSO (SSO) packets, software would perform the following:

1. Software writes out transmit context and data descriptors to the Tx-Queue. See Section 10.5.3.1 for detailed structures.
2. Software internally accumulates the pending amount of posted transmit data and increments a pending packet count.
3. When a quanta needs to be closed (based on the above criteria), software:
    a. Rounds up total amount of pending data to the nearest 64-byte unit.
    b. Updates the current QDR entry and advances software's QDR write pointer.
    c. A transmit doorbell is sent to the device. The doorbell contains the new Tail, the amount of added data from the last doorbell (in 64-byte units rounded up), and the *Quanta Completed* bit is set to one. See Table 10-23.

Software can post a transmit doorbell without closing the current quanta (including for each and every packet) simply by not setting the *Quanta Completed* bit in the TDR doorbell write. Packets transmitted into an otherwise empty transmit queue are considered immediately for transmission based solely on the transmit doorbell contents.

For TSO messages, software would perform the following:

1. Software writes out transmit context and data descriptors to the TDR. See Section 10.5.3.1 for detailed structures.
2. If software has accumulated pending non-TSO (SSO) transmit data and has an open quanta context:

a. Software rounds up total amount of pending data to the nearest 64-byte unit.

b. Software updates the QDR entry and advance write pointer.

c. Hardware automatically closes any previous open quanta on LSO.

3. For TSO, software initializes a quanta descriptor for each TSO message:

   a. Software provides the exact number of bytes in byte-level granularity. Software updates the current QDR entry.

   b. A transmit doorbell is sent to the device. The doorbell format is detailed in Table 10-24.

Note: The minimum supported TSO MSS is 88. Whenever the supplied MSS is less than 256 bytes, the resulting TSO segments generated would be scheduled for transmission as if the MSS was 256 bytes.

- The above applies for both the TSO Quanta Descriptor and the TSO Doorbell (*First Quanta Length*).

In TSO message, a quanta must not span on more than 16 Tx-Descriptors including the context descriptor and header buffers.

10.5.6.3 Head Drop via Quanta Expiration

As software initializes SSO quanta descriptors with the associated transmit data descriptors on the transmit queue, software can specify an expiration timestamp. If the individual packets associated with the transmit quanta end up being selected for transmission later than the given timestamp, the packets are internally dropped by the transmit pipeline. Packets can be delayed for a variety of reasons, such as transient transmit scheduling contention, traffic class pause, or other.

The specific packets that are dropped are explicitly identified by Completion Queue notifications.

Note: Quanta expiration is enabled only for queue type #3 (advanced host interface with no TSO).

10.5.6.4 Head Drop via Drop Request

Software can also determine after the fact, that various packets that have been posted to the transmit queue should be dropped. One example could be either the transmit queue length has grown beyond some software-defined threshold, or other method to track forward progress of packets on a given transmit queue.

In this situation, software can utilize an explicit drop doorbell (or use the drop doorbell command) to signal hardware that one or more quanta be dropped from the head of the transmit queue. As one or more packets can be associated with a quanta, several packets can be dropped at once from the head of the queue. If the affected packets have been selected for transmission, a late drop notification is passed through the transmit pipeline.

Note: Head Drop is enabled only for queue type #3 (advanced host interface with no TSO).

If late packet transmission is an ongoing issue, the situation suggests a system configuration and usage issue. Various statistics can be queried from the various PF instances to determine the source of scheduling contention.

Hardware accumulates the drop requests and executes the drops when a Tx-Queue is scheduled for transmission. The internal drop accumulator is a 7-bit counter per Tx-Queue. It can accumulate up to 127 quantas drop request. Overflow is silently ignored. When the Tx-Queue is empty, any drop request is ignored and the accumulator is reset.

10.5.6.5 Quanta Size Selection

Quanta sizes are used in combination with the various packet scheduler rules to implement packet pacing, while simultaneously interleaving packet transmissions from other queues on the device. For legacy LAN queues with TSO enabled, the device's internal data cache is optimized for 1 KB quanta sizes. Use of larger quanta values reduces the number of simultaneously-active queues that can be in use and still meet device performance objectives.

Quanta values are further governed by the quanta profile referenced in the queue context. The quanta profile places limits on the number of descriptors as well as the number of commands per quanta. These upper bounds ensure a queue (perhaps used by an untrusted VF) cannot be used in such a manner as to deliberately skew the internal queue arbiter to service a queue more frequently than other queues. Attempts to exceed these profile thresholds triggers a malicious driver detection event to the associated PF.

10.5.7 Performance Consideration

Tx path is required to operate 80 MPPS for packets with one data descriptor with no context descriptor, or with context descriptor when the next conditions are true:

- TSO bit is cleared.
- TSYN is cleared or TSYN is disabled in queue context.
- SWICH field != TARGET_VSI

10.5.8 Stateless Transmit Offloads

10.5.8.1 Insert Ethernet CRC Bytes

See Section 7.12.1.1.

10.5.8.2 Insert L2 Tags (VLAN)

See Section 7.12.3.2 for a complete description of L2 tag insertion. This section provides some rules relating to the supported headers for which L2 tag can be inserted by the device. See Figure 10-11 for the structure of the supported packets, and Table 10-40 for the L2 tag insertion.

Table 10-40. Transmit L2 Tag Insertion Offload

Packet Type	Parsing Hints in the Transmit Descriptors	L2 Tag Insertion Supported
Non-tunneled packet	MACLEN = the length of the L2 header provided by the software.	Insert L2 tags depending on IL2TAG1 and IL2TAG2 setting. The L2 tags are inserted after.
NSH over L2 packet	MACLEN = the length of the L2 header provided by the software up to including the NSH header.	
MAC in MAC packet	MACLEN = the length of the 2 x L2 headers provided by the software.	Yes, only for the inner L2 header depending on IL2TAG1 and IL2TAG2 setting. The L2 tags are inserted after.
IP in UDP tunneling or IP in GRE	Same as non-tunneled packet.	

Table 10-40. Transmit L2 Tag Insertion Offload [continued]

Packet Type	Parsing Hints in the Transmit Descriptors	L2 Tag Insertion Supported
MAC in UDP tunneling or MAC in GRE	MACLEN = same as non-tunneled packet.	
	= The length of the tunneling UDP or GRE up to the inner IP header (including optional NSH or MPLS)	Insert VLAN after NATLEN if IL2TAG_IL2H is set. ***Note:*** The IL2TAG2 and IL2TAG_IL2H are mutually exclusive (only one of them can be set). Inner VLAN insertion offload should not be requested if MPLS follows the tunneling MAC header.

10.5.8.3 Transmit L3 and L4 Integrity Offload

The E810 can offload the following L3 and L4 integrity checks:

- IPv4 header(s) checksum for "simple" and tunneled packets.
- Inner TCP or UDP checksum.
- Tunneling UDP checksum.
- SCTP CRC integrity.

If a checksum is required, software should provide it, as well as the inner checksum value(s) that are required for the outer checksum. To request L3 and L4 integrity offloads, software should define the packet format and the required offload in the transmit descriptors as described in Figure 10-11 and Table 10-42.

Some rules for integrity offload are:

- Offloads for inner headers to an IPv6 header with fragment extension header or fragmented IPv4 packets do not make sense. Note that this rule is not enforced by the E810.
- IPv6 support: The pseudo header for the L4 checksum takes into account the addresses in the IPv6 header ignoring the optional extension headers. Packets with Routing Header type 2 or Destination Options Header with Home Address option contain an alternative IP Address in the extension header. Therefore the OS should not request checksum nor segmentation offload for packets with routing extension header type 2 or Destination Options Header with Home Address option.
- IP Header Length requirements (IPLEN and EIPLEN):
  - In case of a non-tunneled packet, the IPLEN defines the IP header length in DWord units, and the IIPT defines the IP header type.
  - In case of tunneled packets, the IPLEN defines the inner IP header length, and the EIPLEN defines the outer (external) IP header length. The IIPT defines the inner IP header type, and the EIPT defines the outer IP header type.
  - The Length field in the IP header is prepared by the software in the packet buffers. This length field includes the payload of the IP header. In MAC tunneling, the inner MAC header (including optional VLAN tag) is part of the outer IP header payload. This optional VLAN tag can be embedded in the packet buffers or by using the IL2TAG_IL2H option in the transmit context descriptor. Regardless of the above, the optional VLAN tag should be taken into account in the length field of the outer IP header.
  - For IPv4, it should be at least 20 bytes (basic header size), and not more than 60 bytes.
  - For IPv6, it should be at least 40 bytes (basic header size), and up to the maximum size enabled by the parsing fields for basic IPv6 header and its extension headers.

- L4 Header Length requirements (L4LEN):
  - For TCP, it should be at least 20 bytes (basic header size), and not more than 60 bytes.
  - For SCTP, it should be set to 12 (SCTP common header size).
  - For UDP, it should be set to 8 (UDP header size).

Table 10-42 lists all supported packet formats and the processed integrity. The table uses the following notations:

- **IP** — A generic term for IPv4 header or IPv6 header. The IPv4 header can have IP option headers and the IPv6 header can have IPv6 extension headers.
- **L4** — A generic term for UDP, TCP, or SCTP headers.
- **IP checksum** — Meaningful only for IPv4.
- **Checksum** — A generic term for UDP and TCP checksum as well as SCTP CRC integrity.

Offload details:

- IPv4 checksum calculation (both inner and outer IP header for tunneled packets):
  - The software should set the IPv4 checksum to zero.
  - The hardware calculates the IPv4 header checksum starting at the beginning of the IPv4 header up to the end of the header.
- Outer UDP checksum calculation (UDP tunneling):
  - The software provides the pseudo IP header checksum in the outer UDP header.
  - The hardware calculates the UDP checksum starting at the beginning of the tunnel header up to the end of the packet, which includes the pseudo header provided by software.
- UDP and TCP checksum calculation (inner L4 header in case of UDP/GRE tunneling):
  - The software provides the pseudo IP header checksum in the L4 header.
  - The hardware calculates the L4 checksum starting at the beginning of the L4 offset up to the end of the packet, which includes the pseudo header provided by software.
- See Table 10-41 for details of the fields that the software device driver should fill when sending a packet.
- SCTP CRC calculation (inner L4 header in case of UDP/GRE tunneling):
  - The software should set the CRC in the header to zero.
  - The hardware calculates the CRC according to SCTP standard starting at the beginning of the SCTP header up to the end of the packet.
- Tunneling UDP/GRE Header Length requirements:
  - For UDP/GRE tunneling, the header can be any value in 2-byte granularity from eight bytes (bare UDP header) and up to 254 bytes (including optional network key and optional inner L2 header up to including the last EtherType). Note that the header length includes only those bytes provided in the packet buffers in host memory.

Table 10-41. Pseudo Header Pre-Loaded Values

Field	Single Send Packet (No TSO)	Transmit Segmentation Offload (TSO)
Outer IP Length	**IPv4**: IP header +payload size **IPv6**: Payload size only, including header extensions. Should match the data size stored in host memory. (Hardware modifies it upon offloads.)	Don't care. (calculated by hardware.)
Outer L4 (UDP) Length	UDP Header size + Payload size Should match the data size that stored in host memories. (Hardware modifies it upon offloads.)	Don't care. (calculated by hardware.)
Outer L4 CS field	IP header pseudo checksum, calculated on IP header fields: **IPv4**: Source addr +Destination addr + Protocol (=0x11) + "UDP Length" **IPv6**: Source addr +Destination addr + NextHeader (=0x11) + "UDP Length" ***Note:*** UDP Length is the payload that follows the IP header. It includes the size of UDP header and data.	IP header pseudo checksum, calculated on IP header fields. Not including "UDP Length": **IPv4**: Source addr +Destination addr + Protocol (=0x11) **IPv6**: Source addr +Destination addr + NextHeader (=0x11)
Inner/Single IP Length	**IPv4**: IP header +payload size **IPv6**: Payload size only, including header extensions. Should match the data size stored in host memory. (Hardware modifies it upon offloads.)	Don't care. (calculated by hardware.)
Inner/Single L4 (UDP/ TCP) Length	Header size + Payload size Should match the data size that stored in host memories. (Hardware will modify it upon offloads.)	Don't care. (calculated by hardware.)
Inner/Single L4 CS Field	IP header pseudo checksum, calculated on IP header fields: **IPv4**: Source addr +Destination addr + Protocol + "L4 Length" **IPv6**: Source addr +Destination addr + NextHeader + "L4 Length" *Protocol* or *NextHeader* values are 0x11 for UDP and 0x6 for TCP. ***Note:*** L4 Length is the payload that follows the IP header. It includes the size of L4 header and data.	IP header pseudo checksum, calculated on IP header fields. Not including "L4 Length": **IPv4**: Source addr +Destination addr + Protocol **IPv6**: Source addr +Destination addr + NextHeader *Protocol* or *NextHeader* values are 0x11 for UDP and 0x6 for TCP.

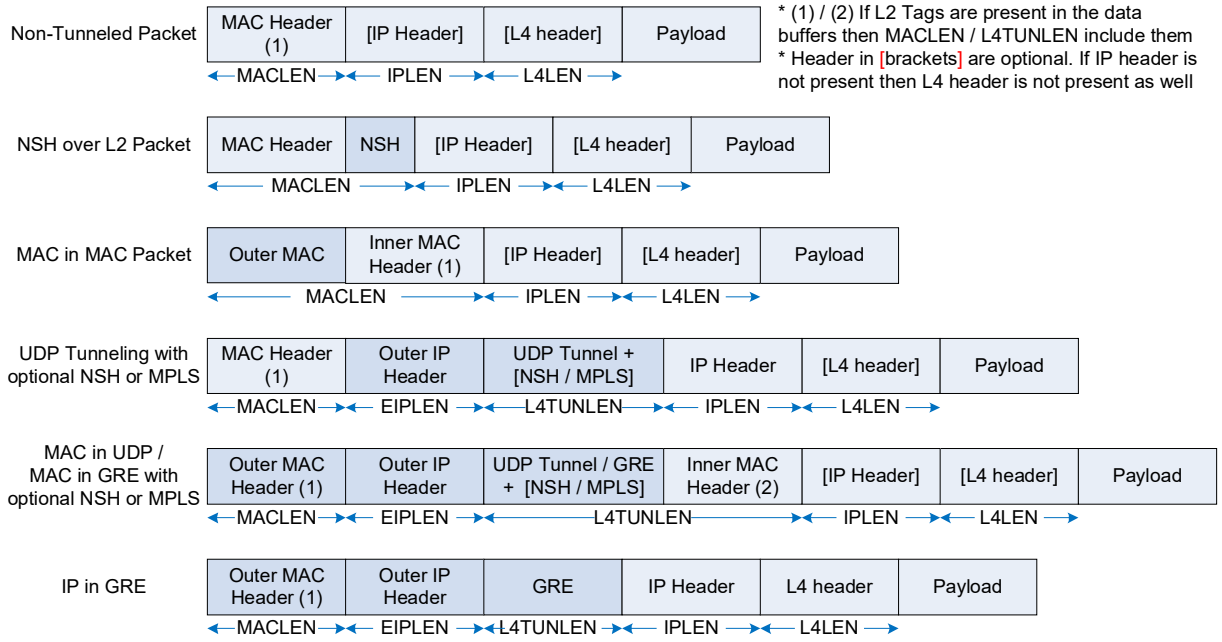


Figure 10-11. Transmit L3 and L4 Header Lengths (in Host Memory) for Integrity Offload

Table 10-42. Transmit Integrity Offload for Packet Types

Packet Type	Parsing Hints and Offload Enablement in the Transmit Descriptors[1,2]	Supported Transmit Checksum Offload
Fragmented IPv4 or IPv4 -> Unknown	IIPT = 10b or 11b. L4T = 00b (unknown).	IPv4 checksum if IIPT = 11b.
Fragmented IPv6 or IPv6 -> Unknown	IIPT = 01b. L4T = 00b (unknown).	None.
IP -> L4	IIPT = 10b or 11b for IP.v4. IIPT = 01b for IPv6 IPLEN = The length of the IP header (including IP optional or extension headers). L4T = 11b, 01b, 10b (UDP, TCP, SCTP). L4LEN = L4 header length. EIPT = EIPLEN = L4TUNT = 0 (no tunneling).	IPv4 checksum if IIPT = 11b: L4 checksum if L4LEN is meaningful (i.e., L4T <> 0).
IP -> (Fragmented IP or IP -> Unknown)	IIPT = 10b or 11b for inner IPv4. IIPT = 01b for inner IPv6. IPLEN = The length of the inner IP header (including IP optional or extension headers). L4T = 00b (unknown). EIPT = 10b or 11b for outer IPv4. EIPT = 01b for outer IPv6. EIPLEN = The length of the outer IP header (including IP optional or extension headers). L4TUNT = 00b (no L4 tunneling).	Inner IPv4 checksum if IIPT = 11b. Outer IPv4 checksum if EIPT = 11b. No L4 checksum.

Table 10-42. Transmit Integrity Offload for Packet Types [continued]

Packet Type	Parsing Hints and Offload Enablement in the Transmit Descriptors[1,2]	Supported Transmit Checksum Offload
IP -> IP -> L4	IIPT = 10b or 11b for inner IPv4. IIPT = 01b for inner IPv6. IPLEN = The length of the inner IP header (including IP optional or extension headers). L4T = 11b, 01b, 10b (UDP, TCP, SCTP). EIPT = 10b or 11b for outer IPv4. EIPT = 01b for outer IPv6. EIPLEN = The length of the outer IP header (including IP optional or extension headers). L4TUNT = 00b (no L4 tunneling).	Inner IPv4 checksum if IIPT = 11b. Outer IPv4 checksum if EIPT = 11b. L4 checksum if L4LEN is meaningful (i.e., L4T <> 0).
IP -> (Tunneling UDP/GRE) -> IP -> L4	IIPT = 10b or 11b for inner IPv4. IIPT = 01b for inner IPv6. IPLEN = The length of the inner IP header (including IP optional or extension headers). L4T = 11b, 01b, 10b (UDP, TCP, SCTP). EIPT = 10b or 11b for outer IPv4. EIPT = 01b for outer IPv6. EIPLEN = The length of the outer IP header (including IP optional or extension headers). L4TUNT = 01b for UDP/GRE tunneling. L4TUNLEN = Tunneling header length.	Same as IP -> IP -> L4.
IP -> (Tunneling UDP/GRE) -> (Fragmented IP or IP -> Unknown)	IIPT = 10b or 11b for inner IPv4. IIPT = 01b for inner IPv6. IPLEN = The length of the inner IP header (including IP optional or extension headers). L4T = 00b (unknown). EIPT = 10b or 11b for outer IPv4. EIPT = 01b for outer IPv6. EIPLEN = The length of the outer IP header (including IP optional or extension headers). L4TUNT = 00b (no L4 tunneling).	Same as IP -> (Fragmented IP or IP -> Unknown).

Table 10-42. Transmit Integrity Offload for Packet Types [continued]

Packet Type	Parsing Hints and Offload Enablement in the Transmit Descriptors[1,2]	Supported Transmit Checksum Offload
IP -> (Tunneling UDP/GRE) -> [NSH/MPLS] -> MAC (w/wo VLAN) -> IP -> L4	IIPT = 10b or 11b for inner IPv4. IIPT = 01b for inner IPv6. IPLEN = The length of the inner IP header (including IP optional or extension headers). L4T = 11b, 01b, 10b (UDP, TCP, SCTP). EIPT = 10b or 11b for outer IPv4. EIPT = 01b for outer IPv6. EIPLEN = The length of the outer IP header (including IP optional or extension headers). L4TUNT = 01b for UDP/GRE tunneling. L4TUNLEN = UDP/GRE header length in the packet buffers up to excluding the inner IP header.	Same as IP -> IP -> L4 plus tunneling UDP checksum.
IP -> (Tunneling UDP/GRE) -> [NSH/MPLS] -> MAC (w/wo VLAN) -> (Fragmented IP or IP -> Unknown)	IIPT = 10b or 11b for inner IPv4. IIPT = 01b for inner IPv6. IPLEN = The length of the inner IP header (including IP optional or extension headers). L4T = 00b (unknown). EIPT = 10b or 11b for outer IPv4. EIPT = 01b for outer IPv6. EIPLEN = The length of the outer IP header (including IP optional or extension headers). L4TUNT = 01b for UDP/GRE tunneling. L4TUNLEN = UDP/GRE header length in the packet buffers up to excluding the inner IP header.	Same as IP -> (Fragmented IP or IP -> Unknown).

1. Common settings to all cases: When context descriptor is used, the TSO flag should be cleared.
2. When IP-IP or other supported tunneling is transmitted, a context descriptor must be used to provide the additional fields types and sizes. For non-tunneled packets, the context descriptor might be needed for other offloads than checksum. The values indicated in this table assume that the context descriptor is used.

10.5.8.4 Transmit Segmentation Offload (Also Known as TSO or LSO)

Transmit Segmentation Offload (TSO, also called Large Send Offload - LSO) enables the TCP/IP or UDP/IP stack to pass a ULP datagram larger than the Maximum Transmit Unit (MTU) size to the network device. The E810 divides the large ULP datagram to multiple segments according to the MTU size, as illustrated in the Figure 10-12. The size of the ULP datagram supported for TSO can be as small as a single byte (obviously transmitted on a single segment) and up to 256 KB (2^18) supporting TSO and “Giant” TSO.

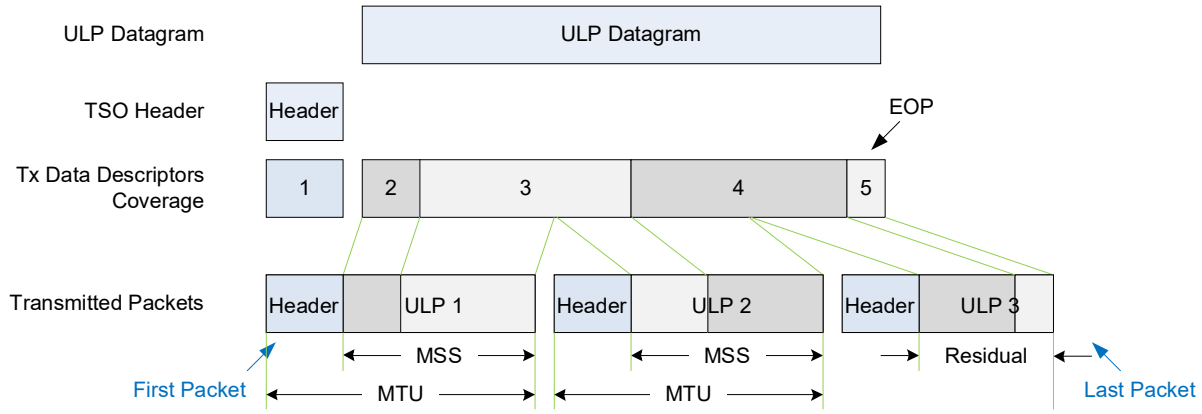


Figure 10-12. TSO Functionality (Example)

10.5.8.4.1 Frame Formats and Assumptions

The following packet formats are supported for TSO:

Note: For all the formats below, IP is IPv4 or IPv6 with/without Option/Extension headers. L4 is TCP or UDP.

- IP -> L4
- IP -> IP -> L4
- IP -> Tunneling header(s) -> IP -> L4
- IP -> Tunneling header(s) -> MAC -> [VLAN] -> IP -> L4

SNAP packet formats are not supported for TSO.

The following assumptions apply to the TCP segmentation implementation in the E810:

- TSO is activated by setting the *TSO* flag in the transmit context descriptor. On top of it, the software should follow the data and context descriptor settings for integrity offload as described in Table 10-42.
- The TSO header (L2, L3 and L4) should be provided by a maximum three descriptors, while still allowed to mix header and data in the last header buffer. The maximum size of the TSO header is 512 bytes.
- The maximum size of a single TSO can be as large as 256 KB minus 1 (defined by the *TLEN* field in the transmit context descriptor).
- The *RS* bit in the data descriptor can be set only on the last data descriptor of the TSO (on which the *EOP* bit is set).
- It is assumed that the software initializes the “pseudo header” checksum excluding the TCP length (as opposed to single send on which the pseudo header checksum includes the TCP length).

10.5.8.4.2 Transmit Segmentation Flow

The TSO flow includes the following steps:

1. The protocol stack receives from an application a ULP datagram to be transmitted.
2. The protocol stack calculates the number of packets required to be transmitted based on the MSS.
3. The stack interfaces with the device driver and passes the block down with the appropriate header information: Ethernet, IP header(s), optional tunneling headers, and the L4 header.
4. The stack interfaces with the device driver and commands the driver to send the whole datagram. The device driver sets up the interface to the hardware (via descriptors) for the segmentation.
5. The hardware fetches the segmentation parameters as well as the data and header buffers description by the transmit context and data descriptors.
6. The hardware fetches the header and data buffers and then transmits them by segments according to the TSO parameters.
7. Dynamic fields set by the software:
   - For IPv4 the IP header, checksum should be set to zero.
   - The *Total Length* field in the IP header(s) should be set to zero.
   - The IP ID of the first segment to be transmitted.
   - The pseudo header checksum of the TCP/UDP header should be calculated and placed as part of the packet data in the TCP or UDP checksum offset.
   - Initial value of the TCP flags (see later on for its value in the TSO packets).
8. Dynamic fields in the IPv4 header(s) that are modified by the hardware:
   - The *Total Length* field should reflect the IP payload size plus the IP header length. The L4 payload size is MSS for all packets but the last one, which contains the rest of the data. So for the inner IP header (in case of tunneling), it is:

     L4 Payload + L4LEN + IPLEN.

     This is the same rule for the IP header in non tunneling case. For an outer IP header (in case of tunneling) it equals:

     L4 Payload + L4LEN + IPLEN + optional L4TUNT length + EIPLEN

     Note that in case of MAC tunneling, the length of the outer IP includes the inner L2 header, including optional VLAN (updated by the hardware if the VLAN is inserted by the hardware).
   - The *Identification* field (in the IP header in non-tunneled packets, or the inner IP header in tunneled packets) is taken from the TSO header in the first segment, and is increased by one for each transmitted segment.
   - The header checksum is calculated after the other parameters in the IP header are updated.
9. Dynamic fields in the IPv6 header(s) that are modified by the hardware:
   - The *Payload Length* should reflect the payload size. It is the MSS for all packets but the last one, which contains the rest of the data. The *Payload Length* should reflect the IP payload size. Therefore, for the inner IP header (in case of tunneling) it is:

     L4 Payload + L4LEN + IP Extensions

This is the same rule for the IP header in non tunneling case. For an outer IP header (in case of tunneling) it equals:

L4 Payload + L4LEN + IPLEN + optional L4TUNT length + IP Extensions

The *IP Extensions* length equals to *IPLEN* minus 40, or *EIPLEN* minus 40 for the inner or external IP headers, respectively. Note that in case of MAC tunneling, the length of the outer IP includes the inner L2 header including optional VLAN.

10. Dynamic fields in the TCP header that are modified by the hardware:
    - The *Sequence* number in the TCP header is taken from the TSO header in the first segment. It is then incremented by MSS for each transmitted segment.
    - The TCP flags are taken from the TSO header. The header is then masked (logic AND) by the *TCPMSKF*, *TCPMSKM*, and *TCPMSKL* fields in the GLLAN_TSOMSK_F/M/L global registers. The *TCPMSKF* is used for the first segment of the TSO, *TCPMSKL* is used for the last segment of the TSO, and *TCPMSKM* is used for all other segments of the TSO. If the TSO is composed of a single segment, it is processed the same as the last segment of a multi-segment TSO.
    - The TCP checksum is calculated starting by the initial value (of the pseudo header). The checksum includes the updated TCP header, TCP payload and the calculated TCP length (equals the payload size plus the TCP header size).
11. Dynamic fields in a tunneling UDP header that are modified by the hardware:
    - The UDP length reflects the size of the tunneling UDP payload plus 8 (which is the size of the UDP header).
    - The UDP checksum is calculated starting by the initial value (of the pseudo header). The checksum includes the updated UDP length and the UDP payload.

10.5.8.4.3 Transmit Arbitration

The E810 arbitrates between transmit queues at packet boundaries while enabling each queue to transmit at least pre-defined quanta (as long as the queue is not empty). TSOs are not an exception to this rule. If the queue exhausts its quanta in the middle of a TSO, the E810 switches to the next queue in line at the end of the transmitted segment of the TSO. See Section 8.3, "Transmit Scheduling" for a description of transmit arbitration.

10.5.8.4.4 Segmentation Indication to the Hardware

Software indicates a TCP.UDP segmentation by a transmit context descriptor just before the data descriptor with the following parameters:

- TSO flag is set, indicating a TSO is requested.
- Insert STag or External VLAN can be set only by the PF (the same as a single send).
- Switch control fields must not be enabled for TSO.
- MSS should be set to the required size of the L4 payload on each segment (MTU minus the size of the headers).
  - MSS smaller than 88-bytes or larger than allowed (declared in field), are considered malicious. The respective queue is stopped and an interrupt is issued to the PF. The relevant event is "Bad LSO MSS".

- *TLEN* is the total ULP datagram length.
- Other parameters in the data descriptor(s) and the context descriptor are defined the same as a single send.

The data descriptors indicate the TSO header as well as the ULP datagram while following the frame formats and assumptions described in Section 10.5.8.4.1.

Chapter 11 Protocol Engine

11.1 Protocol Engine Overview

The Protocol Engine (PE) adds Remote Direct Memory Access (RDMA) capabilities to the traditional LAN functionality found in standard NICs. RDMA is a *networking performance optimization* that enables servers to communicate across a network using high-performance, low-latency, zero-copy DMA semantics. It is designed to reduce host CPU utilization, host memory bandwidth used for network traffic, and network latency when compared to traditional networking stacks such as sockets with TCP/IP. Here are some of the RDMA capabilities provided by the PE:

- Secure, direct access to PE hardware for userspace or kernel applications
  - — Supported in Virtual Machine environments using SR-IOV.
- Translation Protection Table (TPT)
  - — Similar to a CPU Memory Management Unit (MMU).
  - — Enables definition of > 1M Memory Regions.
  - — A Memory Region is a virtually or logically contiguous area of application address space registered with the OS. An enabled Memory Region enables the PE to perform DMA access for local and (optional) remote requests, and enables userspace applications to specify buffers in Virtual Address space.
  - — 100M+ Virtual-to-Physical page translations.
- Reliable Connection Transport
  - — 100K+ connections, analogous to a DMA engine with 100K+ channels
  - — Sequence number checks
  - — Acknowledgment
  - — Dynamic congestion control
  - — Retransmission
  - — Timers
- IP Routable RDMA

The remaining subsections in this chapter are organized as follows:

- Section 11.2, “Features” — Gives details on PE features.
- Section 11.3, “Functional Description” — Describes the hardware infrastructure external to PE that a silicon device must implement, in order to successfully integrate PE.
- Section 11.4, “Verbs Programming Model” — Defines PE implementation details related to the RDMA *Verbs* programming model.
- Section 11.5, “Resource Management” — Defines how PE hardware resources are managed by a driver.
- Section 11.6, “RDMA Functionality” — Defines RDMA-related structures a driver must program.
- Section 11.7, “UD/UDA Functionality” — Defines UD/UDA functionality.

- Section 11.8, "UDA Functionality" — Describes the PE UDA capabilities, a way to give kernel applications direct access to PE hardware to enable iWARP connection setup and error handling.
- Section 11.9, "Protocol Engine Statistics" — Describes PE hardware statistics counters.
- Section 11.10, "SR-IOV Protocol Engine Functionality" — Describes PE SR-IOV support.
- Section 11.11, "NVM RDMA Register Initialization" — Describes RDMA registers that are initialize from NVM values.

11.2 Features

Table 11-1 shows the Protocol Engine feature set.

Table 11-1. Protocol Engine Features

Category	Description
General	**RDMA Protocol Support** Simultaneous support for both iWARP and RoCE v2.
General	**PCIe Function Enabled for RDMA** All of the device's PCIe Physical Functions, and any 32 of the device's PCIe Virtual Functions can be enabled to use RDMA. The number of VFs enabled for RDMA can be reduced via resource profiles. RDMA resources can be redistributed between VFs and PFs based on resource profiles. ***Note:*** The maximum number of Physical Functions supported is 8.
General	**Ethernet Ports Supported** Support for 4 Ethernet ports with 8 TCs, or 8 Ethernet ports with 4 TCs.
General	**IP Version** Support both IPv4 and IPv6 offloaded RDMA connections.
General	**ARP Table** Supports a unique ARP Table instance with up to 65536 entries per PCI function (PF or VF). Each entry contains a 6-byte Ethernet address, plus control/status information for neighbor reachability detection. The ARP Table supplies a destination Ethernet address for all transmitted RDMA packets.
General	**RDMA-Enabled MAC Addresses** The Source Ethernet MAC Addresses are kept with the QP context. There is no table of MAC Addresses.
General	**IP Datagrams/Fragmentation** RDMA IP datagrams are never transmitted with fragmentation. RDMA IPv4 datagrams are always transmitted with the *Don't Fragment* flag set and the *Fragment Offset* field set to 0. Received RDMA IP datagram fragments are sent to the LAN stack and not processed by the RDMA offload engine.
General	**RDMA Statistics** 128 sets of RDMA statistics. Each set can be assigned to an arbitrary/programmable group of one or more Queue Pairs. Any Queue Pair can be assigned to only one stat set. A typical configuration assigns one set to each active PCIe PF and VF, with the extra sets available for assignment as requested by the OS. For example, if 8 PCIe PFs and 32 PCIe VFs are active, then 40 RDMA stat sets are assigned to these functions, with the sets remaining for allocation as requested by the OS.
QP	**Max QP Count** Supports 256K Queue Pairs for RDMA that are dynamically assigned to PCI functions at runtime by Protocol Engine firmware.

Table 11-1. Protocol Engine Features [continued]

Category	Description
QP	**Work Queue Elements (WQEs)** These WQE properties are supported: • Similar WQE format for SQs and RQs. • Each iWARP or RoCE v2 WQE can vary in size from 1 to 14 fragments, with these sizes: 32B, 64B, 96B, 128B, 160B, 192B, 224B, 256B. • Fragments are specified with virtual addresses and are virtually contiguous. • The Protocol Engine performs virtual-to-physical translation using its built-in Memory Management Unit (MMU). • An iWARP or RoCE v2 SQ WQE can (optionally) directly convey 224B of inline data for small message latency optimization. • Send Queue Push mode support is required.
QP	**Work Queues (WQs)** RDMA WQs are mapped into host memory as physically or virtually contiguous ring buffers. Maximum WQ size is 16K entries x 32B = 512 KB. Maximum WQ depth is configurable on a per-WQ basis. Supported values range from 4 to 16K 32B WQEs, in power-of-two increments. Dynamic WQ resizing is not supported.
QP	**SQ Operations (Baseline)** Support for these baseline SQ operations: • Send • Send with Invalidate • Send with Solicited Event • Send with Solicited Event and Invalidate • RDMA Write • RDMA Read (iWARP-style single local SGE) • RDMA Read with Local Invalidate (iWARP-style single local SGE) • Bind Memory Window • Fast-Register Memory Region • Invalidate Local STag. These operations work with either iWARP or RoCE v2, except the iWARP-style RDMA Reads, which are iWARP-only.
QP	**SQ Operations (RDMA Write with Immediate Data)** Support for RDMA Write with Immediate Data. CQE and SQ WQE formats enable 8B of Immediate Data for both iWARP and RoCE. For iWARP, 8B of Immediate Data is conveyed on the network. For RoCE, only 4B of Immediate Data is conveyed on the network.
QP	**SQ Operations (RDMA Read Special)** Support RDMA Read (IB-style with multiple local SGEs). For RoCE v2, this is the only form of RDMA Read. Each iWARP QP is configured by software to use either canonical iWARP-style RDMA Reads, or to use IB-style RDMA Reads. In the later mode, RDMA Read with Local Invalidate must not be considered an illegal operation, for backwards compatibility with existing iWARP applications.
QP	**SQ Operations (Send with Immediate)** For RoCE v2 Queue Pairs only, support Send with Imm, Send with SE and Imm.
QP	**Number of RDMA Reads** The number of outstanding inbound RDMA Reads, defined by Inbound RDMA Read Queue Depth (IRD) is configured independently per QP. These IRD settings are supported: 2, 8, 32, 64, 128, 256. IRD can not be modified after the QP has been created. The number of outstanding outbound RDMA Reads, defined by Outbound RDMA Read Queue Depth (ORD), is also configured independently per QP. These ORD settings are supported: 0 to 255, in single-step increments. ORD can be modified after the QP has been created, if the proper quiesce conditions are met.
QP	**Send Queue Push Mode** This device supports Send Queue *Push Mode*. In this mode, the Host CPU writes or pushes SQ WQEs with or without inline data to the device memory-mapped address space using CPU write-combining buffers. Each of the device's PCIe PFs MUST support up to 1024 separate 4KB Push Pages. Each of the device's RDMA-enabled VFs MUST support 16 4KB Push Pages. The Push Pages are typically exposed through an extension of the memory BAR that contains the CSRs. PF Push Pages MUST be assignable to VMs via device firmware in para-virtualized driver models.

Table 11-1. Protocol Engine Features [continued]

Category	Description
CQ	**Completion Queues** Supports up to 524288 CQs that are dynamically assigned to PCI functions at runtime by device firmware. A CQ/ring buffer can be either virtually or physically contiguous. CQE size is 32B for standard operations, and 64B for special cases including some Immediate Data and UD operations. 32B CQEs can optionally be padded to 64B cache-line boundary to avoid memory conflicts. Max CQ size is 1M entries x 64B = 64 MB. Supports user-defined mapping of WQs to CQs. Supports CQ resizing, CQ size can be increased or decreased while the CQ is active. Supports CQ overflow detection.
EQ	**Completion Event Queues** Supports 256 Completion Event Queues for RDMA that are dynamically assigned to PCI functions at runtime by device firmware.
EQ	**Asynchronous Event Queues** Supports 48 Asynchronous Event Queues for RDMA that are dynamically assigned to PCI functions at runtime by Protocol Engine firmware.
MMU	**Protection Domains** Each PCIe PF or VF enabled to use RDMA can define up to 256K Protection Domains.
MMU	**Memory Regions and Windows** Each PF or VF enabled to use the Protocol Engine can allocate up to 4M (i.e. 2^{22}) Memory Regions or Memory Windows. The maximum size of a standard Memory Region or Memory Window is 32TB (245 bytes). The Protocol Engine also supports a special privileged unbounded Memory Region configuration. When set to unbounded, a Memory Region is configured to register all of contiguous physical memory, and to treat Memory Region Length as "unused and unenforced".
MMU	**Wide Memory Windows** Support Wide Memory Windows for both iWARP and RoCE. Wide Memory Windows have Protection Domain scope, whereas standard iWARP Memory Windows have QP scope.
MMU	**Host Memory Page Sizes (Baseline)** Host Memory page sizes supported: 4 KB, 2 MB. Each Memory Region can be independently configured for either page size.
MMU	**Host Memory Page Sizes (Enhanced)** Includes everything in baseline, plus support for 1 GB Host Memory page size.
MMU	**Physical Buffer List (PBL)** Each RDMA Memory Region can be Physically Mapped (that is, the region is physically contiguous in Host Memory) or Virtually Mapped with a one- or two-level PBL. Each RDMA-enabled PF or VF can allocate up to 256M (2^{28}) PBL Entries. A single PBL Entry maps a single Host Memory page.
MMU	**Maximum Virtually Mapped Memory (Enhanced)** The maximum virtually-mapped memory a single PF or VF can register for RDMA depends on host page sizes used. The examples below illustrate this for different page sizes. Supported page sizes are defined elsewhere, and some of these examples might not be relevant for this device. If a PF or VF allocates its maximum limit of 256M PBL Entries, then... ...its maximum virtually mapped memory using 100% 1 GB host memory pages is: 2^{30} x 2^{28} = 2^{58} bytes = 256 PB ...its maximum virtually mapped memory using 100% 2 MB host memory pages is: 2^{21} x 2^{28} = 2^{49} bytes = 512 TB ...its maximum virtually mapped memory using 100% 4 KB host memory pages is: 2^{12} x 2^{28} = 2^{40} bytes = 1 TB
UD	**Address Handles** Supports 128K Address Handles per PCIe function.
iWARP	**Standards Compliance** IETF RFC 5040, 5041, 5042, 5044, 6580, 6581, 7306 (immediate data only, atomics are not supported) RDMA Consortium Verbs
iWARP	**Receive Window Size** Each RDMA connection has a configurable Receive Window with maximum size of 1GB-1B.
iWARP	**Non-Permissive IETF RNIC** Using a term coined in IETF RFC 5044, this device is a non-permissive IETF RNIC (an RNIC that implements the IETF protocols, but not the RDMAC protocols).

Table 11-1. Protocol Engine Features [continued]

Category	Description
iWARP	**MPA - Baseline Support** Supports all of the following: • Insertion of Transmit markers (can be enabled/disabled per QP). • MPA CRC generation for outbound iWARP packets. • Transmit up to four FPDUs in a single Ethernet packet. • Able to transmit partial FPDUs when there is outstanding (unacknowledged) data and an MSS change occurs. • MPA CRC checking on inbound packets (can be enabled/disabled per QP). • Process received Ethernet packets with any number of iWARP FPDUs (bounded by max packet size). • Support for detection/handling of received partial FPDUs. • Drop out-of-order received FPDUs without ACKing the TCP segment containing them.
iWARP	**MPA - Place Out-of-Order Received FPDUs** When Receive Markers are enabled, place out-of-order FPDUs.
RoCE	**Standards Compliance** InfiniBand Architecture Specification Volume 1 Release 1.3 (support the subset of this spec that applies to RoCE v2) InfiniBand Architecture Specification Annex A17: RoCE v2 Sept 2014

11.3 Functional Description

11.3.1 Packet Classification and the PE

The E810 provides several filtering checks that can be used to distinguish LAN traffic that should be handled by the PE from other LAN traffic. The mechanism to direct a packet to be processed by the PE is to enable the RDMA Packet Profiles for RDMA in the VSI. The internal switching components described in Section 7.10, including this special setting, are used to identify a Virtual Station Interface (VSI). Once the VSI has been identified, the checks shown in Figure 11-1 determine if a packet is processed by the PE.

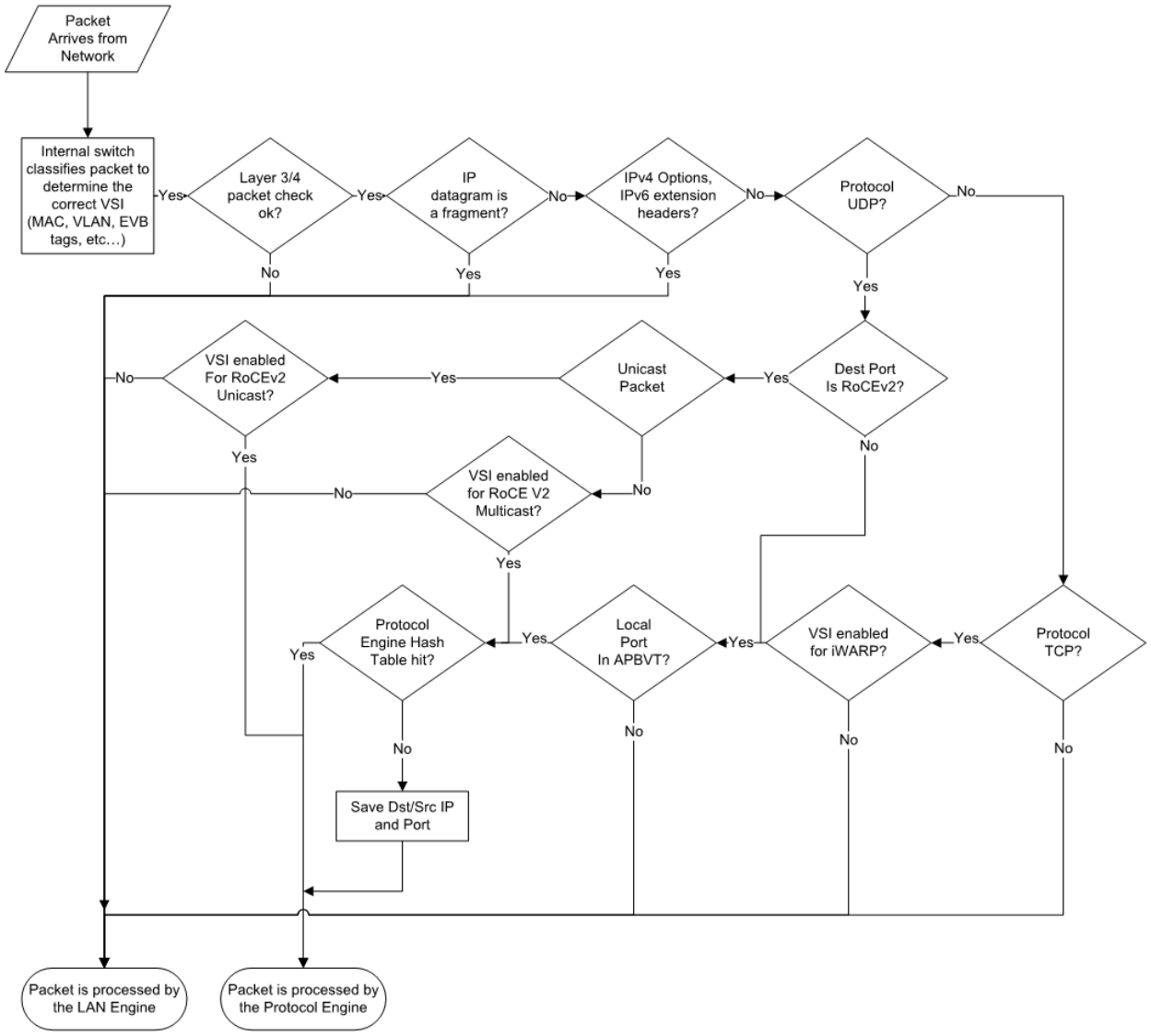


Figure 11-1. PE Packet Classification

Once the internal switch packet classification has been performed, the following checks must be passed before a packet is processed by the PE:

1. The VSI configuration option that enables traffic to be routed to the PE must be enabled.
2. The packet must be a protocol that is supported by the PE. The current protocols that the PE is capable of handling are TCP/IP and UDP/IP. Multicast and unicast packets are supported for UDP/IP.
3. Additional checks to validate the IP and TCP or UDP headers must succeed.
4. If the packet is an IP packet it must not be a fragmented IP datagram.
5. If the packet is targeted for the RoCEv2 port and is a unicast packet, it is sent to the Protocol Engine.

6. If the packet is targeted for the RoCEv2 port and is a multicast packet, it skips the next step. That is, it does not have an APBVT entry but it does go through the Protocol Engine Hash Table.
7. Otherwise, the destination port of the TCP or unicast UDP packet header must have its associated bit set in the Accelerated Port Bit Vector Table (APBVT). APBVT is described in Section 11.5.3.18.
8. The packet is then sent through a hash table lookup and must be found. The fields from the packet that are used for the lookup vary based on the packet contents. TCP/IP packets that do not have the *SYN* bit set, or have the *SYN* bit set and the *ACK* bit set, use the combination of destination MAC Address, VLAN tag (if present), source and destination IP Addresses, and the source and destination port fields. TCP/IP packets with the *SYN* bit set and the *ACK* bit clear or unicast UDP/IP packets use the destination MAC Address, VLAN tag (if present), destination IP Address and destination port. See Section 11.5.3.20 for more details on the PE hash filter. If this check fails, a bit indicating that a hit in the APBVT was found, the IP table index from check number #7 and destination TCP port number are reported in the receive descriptor as status.

7. Finally, the packet must not have IPv4 options or IPv6 extension headers.

If any of the previous checks fail, the packet is processed by the LAN engine.

The PE has some limitations regarding the packet formats that are supported with RDMA and UDA traffic. The PE supports all packet formats that are configured using the internal switch and configured for the VSI. Additional headers (such as an extra inserted IP header, IP and NAT tunneling headers, or IP and Teredo headers) that are inserted for L2 traffic on a per packet basis are not supported for PE traffic. When the VSI is not configured for handling specific header formats, the PE is capable of handling packets with up to one extra packet headers to match the L2 VLAN capability. The VLAN tag specified in QP context matches the definition for L2TAG1 for LAN descriptors.

Two different values have been used for the RoCEv2 well-known port number: 1021 (early development) and 4791 (assigned by IANA). Two port numbers per physical port are recognized by the Parse Graph as RoCEv2 well-known ports. See the section on Packet Parsing for more information on how to program the parser for these values.

11.4 Verbs Programming Model

The verbs programming model is specified for iWARP from the RDMA Consortium.

http://www.rdmaconsortium.org/home/draft-hilland-iwarp-verbs-v1.0-RDMAC.pdf

The E810 supports the verbs constructs of QPs, CQs, and events. The E810 supports verbs events by implementing CEQs and AEQs. Further description of each of these constructs in provided in the following sections. Additional information on the verbs programming model can be found in the link to the RDMA consortium website previously provided.

11.4.1 Verbs from a System View

The following sections provide an overview of each verbs construct supported by the E810 from a system view. It is expected that the user is generally familiar with the verbs programming interface.

11.4.1.1 Asynchronous Event Queue (AEQ)

The E810 supports a single AEQ per PE-enabled PCI function. AEQs are used to report status and errors associated with PE QPs, CQs, and ARP table entries.

AEQs are a packed array of AEQ entries (see Section 11.4.6 for the format of an AEQ entry) located in a virtually contiguous buffer in host memory (see Figure 11-2). AEQs should be sized to enable a two entries for every QP, CQ, and ARP table entry that is active for the PCI function. Each QP, CQ and ARP table entry ensures that it generates only a single AEQ entry at a time.

Software interaction is required for each resource to enable subsequent AEQ entries. If the AEQ is not sized appropriately, AEQ overflows can result [(Head+2)%AEQ_Size=Tail], in which case AEQ entries are lost, and the *AEQE_Overflow* bit in the AEQE is set to notify software that the overflow condition occurred. Once the AEQ has overflowed, no new AEs are delivered to that queue. The AEQ must be destroyed and recreated to resume AE processing.

The initial condition for software shown in Figure 11-2 is *AEQE_Index* set to 0b. The last valid is shown strictly for discussion purposes. The E810 first writes to the AEQE index specified by *Head*. After an AEQE is written, an interrupt is generated if AEQ interrupts are not masked. Once an interrupt has been received that indicates that a new AEQ element (AEQE) is available, software reads the AEQE at *AEQE_Index* and increments *AEQE_Index*.

Software processes all valid AEQEs until it encounters an invalid entry, and stores the index of the invalid entry in the *AEQE_Index* variable. Subsequent AEQ entries might be generated by the E810 after the entry that caused the interrupt while interrupts are masked. For each valid AEQ entry found, the PFPE_AEQALLOC register (see Section 13.2.2.28.10) must be written to notify the E810 that the AEQ entry is available for use by hardware. Writing the PFPE_AEQALLOC register causes the E810 to increment the on-chip tail context variable. The PFPE_AEQALLOC register supports batching of AEQ entry acknowledgment into a single write to enable software to minimize the number of register writes necessary to complete AEQ interrupt processing. PE enabled VF must use the VFPE_AEQALLOC registers instead of the PFPE_AEQALLOC.

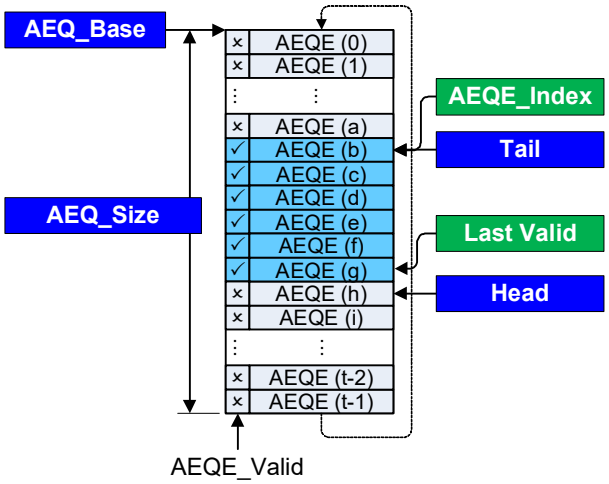


Figure 11-2. AEQ

If software stops processing the AEQ before it has consumed all valid AEQEs, software must use a software-initiated interrupt to return to processing AEQEs. Otherwise, no further interrupts are generated until a new AE is generated. Once the E810 has run through the AEQ and wrapped back to AEQ0, the polarity of the *AEQE_Valid* bit is switched to avoid the need for software to go back and clear the *AEQE_Valid* bit for each AEQE processed.

11.4.1.2 Completion Event Queue (CEQ)

The E810 supports one CEQ per MSI-X vector per PE enabled PCI function and a maximum of 768 CEQs total. HMC resource profiles are used to distribute the number of CEQs across the PE enabled PCI functions.

Software can determine the number of CEQs and the specific instances assigned to a particular PCI function by reading the GLHMC_CEQPART[n] (see Section 13.2.2.20.27) or GLHMC_VFCEQPART[n] (see Section 13.2.2.20.25) registers after the HMC profile has been selected. Each CEQ is associated with a separate interrupt cause. It is expected that the number of CEQs that software uses is the minimum of the number of MSI-X vectors available, the number of CPU cores, and the value reported from the GLHMC_CEQPART[n].*PMCEQSIZE*. Software uses multiple CEQs to distribute the completion process workload across multiple CPUs. Each CQ is individually assigned a CEQ via the CreateCQ or ModifyCQ operations defined in Section 11.5.3.3.

Figure 11-3 shows that CEQs are a packed array of CEQ elements (see Section 11.4.5 for the CEQ element definition) located in a virtually contiguous buffer in host memory. CEQs should be sized according to the maximum number of active CQs that are assigned to the CEQ. Each CQ guarantees that it generates a maximum of one CEQ entry without having software acknowledge that the CEQ entry has been consumed. CEQs are not checked for overflow conditions, so it is important that they are sized correctly or completion events are lost.

The initial conditions for software shown in Figure 11-3 is *CEQE_Index* is set to 0. The last valid is shown strictly for discussion purposes. When the E810 generates a new CE, a CEQE is written to the CEQ at index value of head with the *CEQE_Valid* bit indicating that a new event is available and an interrupt is generated. The E810 bumps *Head* as part of the CEQE generation process, and when the *Head* reaches the end of the CEQ it wraps back to 0. Software management of *CEQE_Index* must match the E810's head algorithm.

Subsequent CEQEs might be written after the entry that caused the interrupt while CEQ interrupts are masked. Software is required to process all valid CEQEs up to the point where the first invalid entry is found. If software stops processing CEQEs before it has found an invalid entry, software must force the E810 to generate a new interrupt using the *SWINT_TRIG* bit in the interrupt control registers. This is necessary since the E810 does not track a tail value for CEQs and therefore cannot determine if a new interrupt is required to process CEQEs that have already been written and not processed by software. Once software has processed a valid CEQ entry, software writes the PFPE_CQACK register (see Section 13.2.2.28.9) to enable the CE to generate new events.

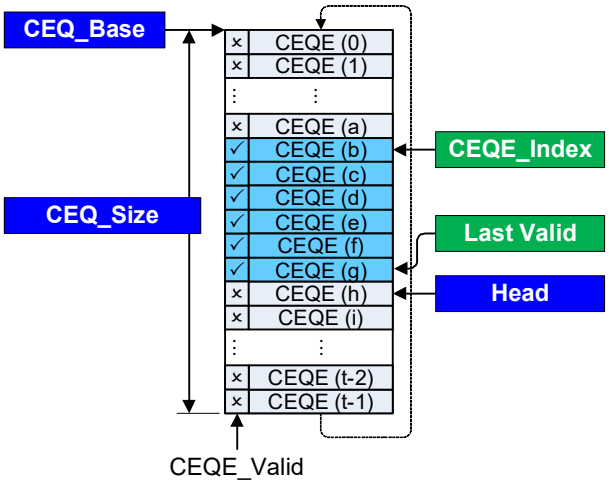


Figure 11-3. CEQ

Once the E810 has wrapped back to CEQE0, the polarity of the *CEQE_Valid* bit is switched to avoid the need for software to go back and clear the *CEQE_Valid* bit for each CEQE processed.

11.4.1.3 Completion Queues

When work requests submitted to Work Queues (WQs) complete, the E810 might post CQEs to associated CQs. Queue Pair WQs might be Send Queues (SQs) or Receive Queues (RQs). SQs and RQs are associated with their CQs at QP creation time. This association remains (and cannot be changed) until the QP is destroyed. Each SQ and RQ can be bound to same or separate CQs. CQs can also be shared by multiple WQs from different QPs.

The E810 supports up to 512K CQs that are distributed among that active PCI functions using HMC resource profiles. See Section 9.3.3 for more information on the resource distribution mechanism used with CQs. The E810 maintains the context of each Completion Queue in CQ context data structures in the HMC's function private memory space. The storage elements of CQs reside in system memory. See Section 9.3 for more information on the E810's usage of host memory for CQ context.

As shown in Figure 11-4, CQs are organized as a circular array of CQEs. Each CQE is 32 or 64 bytes in length, and the format is dependent on the type of the WQ that is associated with the CQE. CQP, RDMA, and UDA WQs all can be related to CQs. CQs are managed using the Create/Modify/ DestroyCQ operations defined in Section 11.5.3.3.

In addition to the CQ itself, software maintains a shadow area that the E810 reads when the CQ is getting low on CQEs for hardware to write, or when the CQ has been armed for event generation. The E810 supports the verbs interface calls for requesting completion notification based on the next completion or based on the next completion that is associated with an operation that is a solicited event. The shadow area contains variables that must be maintained by software in an atomic fashion, since the E810 could read that area at anytime. Specifically, the two 32-bit words (first one at byte offset 0 and second at byte offset 32) in the Completion Queue doorbell shadow area must be accessed using atomic 32-bit processor instructions. The *arm_seq_num*, *arm_next* and *arm_next_se* fields are ignored if an arm request has not been made by writing to the PFPE_CQARM register (see Section 13.2.2.28.8). The PFPE_CQARM register exists in CSR space for kernel mode drivers to access and also in the doorbell page area of the PCI function's BAR space that can be mapped directly to a userspace application's memory space for kernel bypass operation. The format of the shadow area for CQs is shown in Table 11-27 on page 1442.

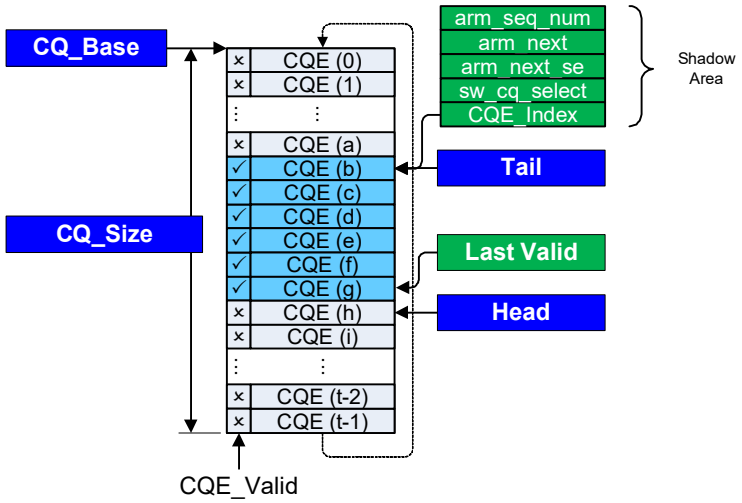


Figure 11-4. CQ

The CQ is organized as a circular queue, written by the E810 and read by the driver software. Initially, both the E810 and the driver software point to the first entry in the queue. The E810 advances the *Head* context field after it writes a CQE to the CQ. Similarly, when the driver picks up a CQE, it advances the *CQE_Index* context field in shadow area in host memory. When *Head* approaches *Tail*, the E810 reads the CQ shadow area and writes the *CQE_Index* value from the shadow area to the *Tail* context variable. If *Head* reaches *Tail*-1 when the E810 attempts to write a CQE, an overflow condition is reported and the CQ is put into the error state. CQs generate a single Asynchronous AE_CQ_OPERATION_ERROR event at this point and any further CQEs that are attempted to be generated by the E810 are lost.

Completion events are generated if the CQ is armed for the appropriate event. The E810 generates completion events under the following conditions assuming the *Tail* does not equal *Head*:

- Any CQE is written (or has been written since the last event) and the CQ is armed for the next completion.
- A CQE is written (or has been written since the last event) that is associated with a Solicited Event operation and the CQ is armed for next solicited event.
- The CQ is armed on a recently re-sized CQ and software has not moved over to the new CQ yet based on the *sw_cq_select* value in the shadow area.

Completion Events can be deferred if a Completion Event has been generated for a CQ and the PFPE_CQACK register (see Section 13.2.2.28.9) has not been written by software.

The E810 differs slightly from the verbs specification definition of generating completion events in two ways. First, the E810 generates a completion event for a CQ that is armed for next completion without waiting for a new completion to be generated if it appears that CQEs have not been processed by software (Head != Tail after reading the doorbell shadow area). Second, the E810 does not track the exact location of the solicited events that have been generated since the last completion event. The E810 generates a completion event for a solicited event operation if any solicited event completion has been generated since the last completion event was generated, and it appears that CQEs have not been processed by software.

The E810 process for arming CQs for event generation is simply to first write to the appropriate bit in the CQ shadow area to enable either next or next solicited completion notification events, increment *arm_seq_num*, and then write to the PFPE_CQARM register (see Section 13.2.2.28.8). The E810 then reads the shadow area and the CQ context is used to either immediately generate a new completion event if the CQ has unprocessed CQEs remaining, or arm the CQ to generate a new event once a subsequent CQE is written. As previously described, completion events can be deferred under certain circumstances. The E810 maintains a copy of the last *arm_seq_num* value that was read during the last arm request in CQ context. The E810 compares the value of *arm_seq_num* in CQ shadow area with the value in CQ context during arm requests and drops arm requests that have the same value in the shadow area and in CQ context. This comparison prevents CQ arm requests from rogue applications from changing the arm state of a CQ unless the application also has access to the CQ shadow area.

CQ resize operations with the E810 involve four steps.

1. Allocate a new CQ in host memory based on the new size requested by an application.
2. Issue a ModifyCQ operation to the E810. ModifyCQ operations notify The E810 to start using the new CQ for new CQEs.
3. Completely process CQEs from the old CQ.
4. Start to process CQEs from the new CQ after freeing the buffer for the old CQ. The old CQ can be considered to be completely processed when an invalid CQE has been found in the old CQ and at least a single valid CQE has been encountered on the new CQ.

When the transition to updating *CQE_Index* based on the new CQ occurs, *sw_cq_select* must be incremented. During the ModifyCQ operation, the E810 incremented its *cq_select* value and switched over to writing CQEs to the new CQ. Since there could be pending CQEs still on the old CQ, the E810 compares the CQ shadow area *sw_cq_select* to the hardware *cq_select* value during any reads of the CQ shadow area due to arms or CQ tail updates. If hardware *cq_select* does not match *sw_cq_select*, the E810 ignores *CQE_Index* and does not update CQ *Tail*, assuming that software is still working with the old CQ. Arm requests generate new events immediately, since the E810 is no longer aware of the state of the old CQ. Once software has properly set *CQE_Index* to reflect progress on the new CQ, it must also increment *sw_cq_select* so the E810 will start processing arm requests and CQ *Tail* updates based on the new CQ.

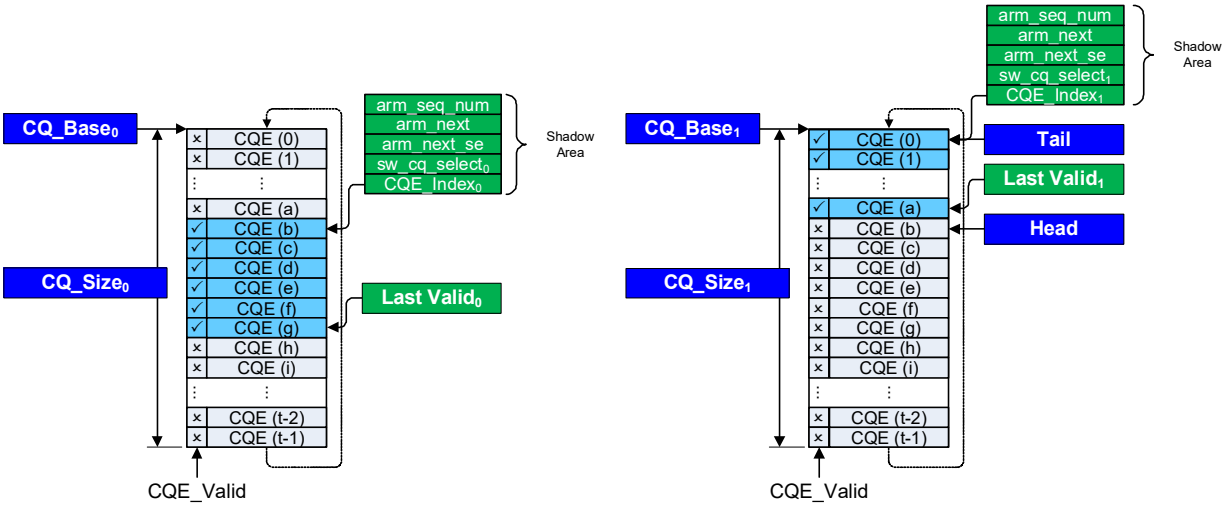


Figure 11-5. CQ Resize Operation

Figure 11-5 shows a CQ during a resize operation. The original CQ is shown on the left side of the diagram. At the time that the ModifyCQ operation occurred for the resize, the CQ had valid CQEs that had not been processed by software. The E810 switched over to the new CQ shown on the right side of the diagram at the time of the ModifyCQ and incremented its *cq_select* value. New CQEs are shown as generated on the new CQ in Figure 11-5. Any arm requests immediately generate new CEs while *sw_cq_select* does no match the hardware *cq_select* value. If the doorbell region is read during this time, the CQE_Index_0 value is ignored because $sw\_cq\_select_0$ does not match *cq_select*. Once software has found an invalid entry in the original CQ and CQE0 from the new CQ is valid, software can reset *CQE_Index* to 0, increment *sw_cq_select*, and start processing CQEs on the new CQ. At this time it is safe to free the memory used for the original CQ, since the E810 no longer writes to the old CQ. While two CQ shadow areas are shown in Figure 11-5, there is really only one. Two are shown to indicated that values of *sw_cq_select* and *CQE_Index* have changed to signal the change over to the new CQ.

11.4.1.4 Memory Registration (Translation/Protection)

In today's modern operating systems, applications running at user protection level use virtual addresses and they are not aware of their data buffers' physical addresses. However, traditional I/O devices require physical addresses to transfer data to/from system memory. This gap is typically bridged by the device driver, which runs in the kernel protection level. Transitioning from user to kernel protection level, and than back to user protection level is not only quite expensive (in terms of MIPs), but also contributes to increased latency. The RDMA architecture solves this inefficiency by:

- Using virtual data buffer addresses in the RNIC programming model.
- Servicing latency critical functions through libraries running at the user protection level

This approach requires RNICs to implement robust address translation and protection techniques. This section provides in-depth descriptions of the address translation and protection mechanism used by the E810. However, the reader is assumed to be familiar with the memory management concepts, operations, and related terms described by the RDMA specification.

The E810 uses a Memory Region Table (MRT) and Physical Buffer Lists (PBLs) to represent virtually contiguous system memory locations that are accessible for RDMA operations. The data structures in MRT are used to describe these memory locations and their protection attributes. PBLs provide the page lists that back each MRT entry. Both of the data structures are HMC objects that are described further in Section 9.3. These concepts, supporting data structures, and related operations are described in Section 11.4.1.4.1. In these sections, the actual MRT and PBL construction details through the HMC are intentionally omitted and a virtually contiguous HMC function private memory space is assumed.

11.4.1.4.1 Address Translation and Protection Overview

As shown in Figure 11-6, the E810 supports a two-level protection and translation flow. The STag and tagged offset pair define the protection attributes and physical address of the system memory being accessed.

The STag portion of the address is used to locate the Memory Region Table Entry (MRTE) in the MRT. This data structure provides the protection attributes (Type, Access Control, Protection Domain and Key), the bounds check attributes (Base Tagged Offset, Length and First Byte Offset), and the address translation attributes (Page Size, and Base PBL Index). These attributes are described in detail later in this section.

After the protection attributes are checked and the address bounds checks are performed, the address translation attributes, along with tagged offset, are used to calculate the PBL virtual address. The translation flow is explained in detail later in this section. The PBL Index portion of the PBL virtual address is used to locate the Physical Buffer List Entry (PBLE), which provides the physical page address. Finally, the page address is concatenated with the offset portion of the PBL virtual address to form the physical system address.

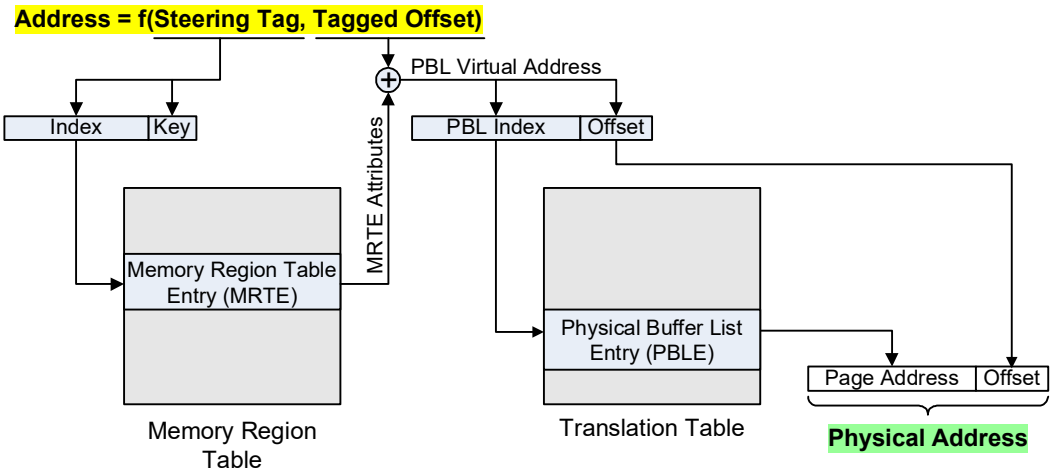


Figure 11-6. STag Decomposition to Physical Address

Depending on its size, each MRTE might be associated with multiple PBLEs. In this case, the PBLEs must be allocated consecutively to form the PBL. The PBLs can also be shared between multiple MRTEs when memory windows or shared memory regions are used. This enables efficient representation of a section of the system memory that is registered multiple times; potentially with different protection attributes.

The E810 also supports two other modes of address translation. The first is termed direct page translation, which can be used when the STag is backed by a physically contiguous buffer is depicted in Figure 11-7. In this case, the PBL translation step is skipped, which enables more efficient calculation of the physical address accessed for a given STag.

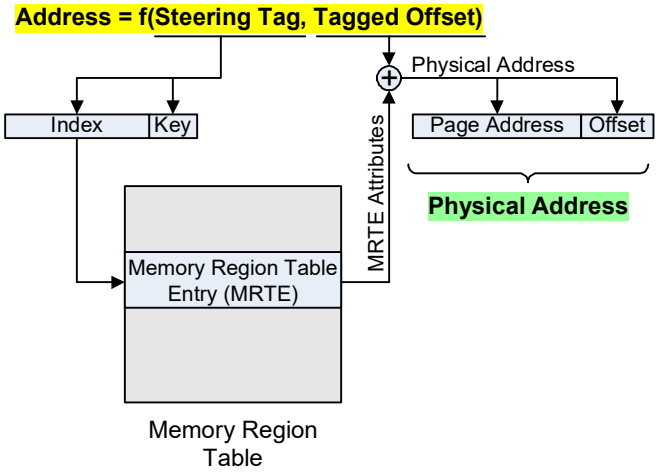


Figure 11-7. Direct Page Translation

The second additional mode of address translation is designed to enable software to make more efficient use of PBL address space when large numbers of large buffers are used for an application at the expense of additional accesses to host memory in order to calculate the ultimate physical address. Figure 11-8 shows a mode where PBLs are accessed twice by breaking the PBL virtual address into three pieces instead of two. The root PBLE is first accessed to find the PBLE associated with the leaf PBL. Once the leaf PBL is identified, the leaf PBL index is used to identify the leaf PBLE address that contains the page address portion of the physical address. In this mode, the leaf PBLs must be either 256 (64 root PBLEs or 32 leaf PBLEs) or 4096 (1024 root PBLEs or 512 leaf PBLEs) bytes in size. The root PBLs can be any size. This mode enables software to slice up the PBL space into fixed size PBLs for large memory regions of up to 2 GB/s instead of having to reserve large contiguous regions of PBL space for a single STag.

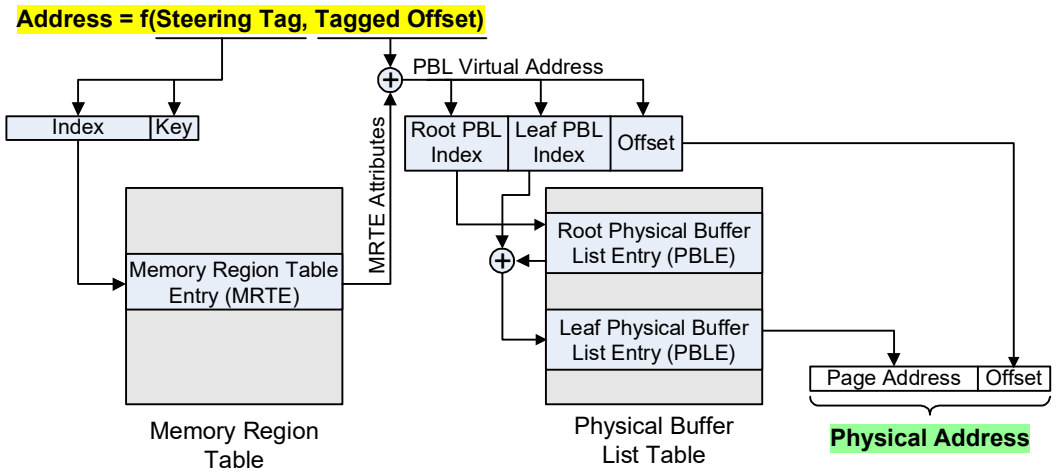


Figure 11-8. Two-Level Page Translation

Each Root PBLE is just a 28-bit object index of the starting PBLE object for the associated leaf PBLE. In other words, each root PBLE points to a another PBLE HMC object that is the start of the leaf PBL. Each leaf PBL is an array of 32 or 512 PBLEs. The size of the root PBL is determined from the region size specified in the MRTE. Each leaf PBLE is a 64-bit pointer to a host memory page which can be a 4 KB or a 2 MB page. For 4 KB pages the lower 12 bits of the host memory page pointer are ignored and for 2 MB pages the lower 21 bits are ignored.

Before we dive into the details of the MRTE and PBL data structures and operations, a brief overview of commonly used terms are provided in the sections that follow.

11.4.1.4.1.1 STag

All local and remote memory accesses require use of an STag. The STag, along with a tagged offset, is used to identify a memory location within a specific memory region or memory window. STags are found in RQ and SQ WQEs and in tagged iWARP messages received from the wire. As shown in Figure 11-9, 32-bit wide STag is further divided into two fields.

Figure 11-9. STag Format

Where:

- **STag Index** — The most significant 24-bits of STag is called the STag Index. Since this field is used as an index into the MRT, it is also known as the memory region table index or MRT index in short. The E810 supports up to 4 MB protection entries. Note that the unused most significant bits of the STag Index can be randomized by software to provided reduced predictability for any MRT attacks.
- **STag Key** — The least significant 8-bits of the STag is called the STag Key. The STag Key field is a user- or driver-provided key that provides an additional level of security for the STag protection check.

11.4.1.4.1.2 Memory Region and Memory Window

Applications must register memory regions prior to accessing the system memory either locally or remotely. Memory regions, and their translation and protection attributes are represented by properly formatted MRT entries. MRT entries of this type are also called Region Entries or RE in short. Memory region registration and de-registration requests are communicated to the E810 (1) by using CQP commands (like: register memory region and de-register memory region) or (2) by work request through a SQ (like: fast register non-shared memory region and invalidate STag).

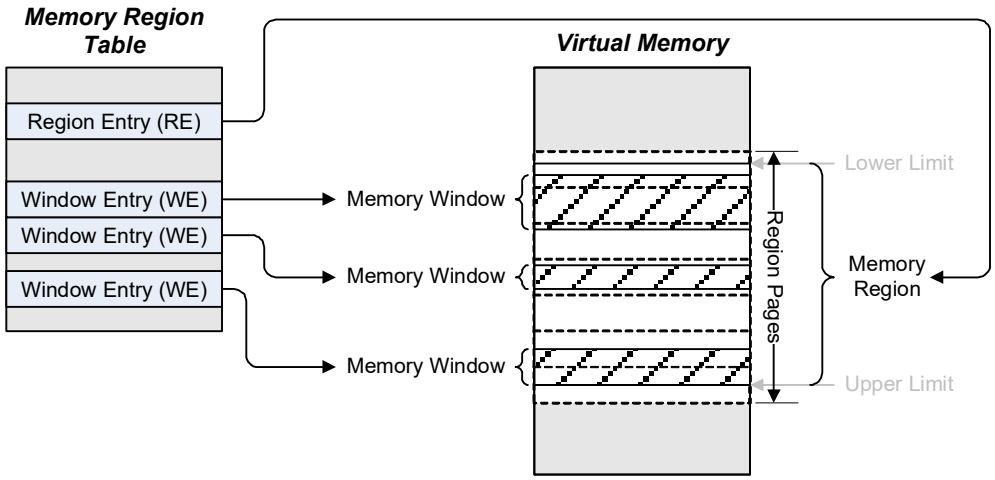


Figure 11-10. Memory Region to Memory Window Relationship

The Verbs specification also defines the memory windows concept. The InfiniBand specification defines Type 1 and Type 2 Windows. Memory windows can be placed anywhere within a valid memory region. Multiple memory windows are allowed within a memory region. Memory windows are allowed to overlap, or be completely included within another memory window. However, memory windows are not allowed to cross memory region boundaries. The E810 supports memory windows through another type of protection entry. These protection entries are also referred as Window Entry (WE). Figure 11-10 shows the relations between memory regions and memory windows.

Memory windows can only be bound to bindable memory regions. A memory window's protection attributes are constrained by the protection attributes of the memory region. The E810 supports byte-level lower and upper address limits for both memory regions and memory windows.

11.4.1.4.1.3 Tagged Offset (TO)

The 64-bit wide Tagged Offset (TO) field specifies the first byte of the buffer being addressed in a memory region or memory window. Memory regions and memory windows have a base attribute as either zero-based TO or Virtual Address (VA)-based TO. For a VA-based TO, the TO of the first memory location associated with the memory region equals the base virtual address value specified when the memory region is registered. For a zero-based TO, the TO of the first memory location associated with the memory region equals zero.

11.4.1.4.1.4 Page Size

Each memory region and memory window has a page size attribute. Memory windows inherit the page size attribute of their associated memory regions. The E810 supports three different page sizes: 4 KB, 2 MB, and 1 GB. All system memory pages that are part of a memory region or memory window must be of the same size.

11.4.1.5 QP

The PE in the E810 supports the verbs QP. QPs are made up of a pair of WQs.

QP WQs are either SQs or RQs. SQs and RQs are associated with their CQs at QP creation time. The E810 supports QPs for PE administrative commands (control QP or CQP), RDMA is supported by iWARP, RoCEv2, and UDA. Each QP type has unique operation types that are described in Section 11.5.3, Section 11.6.6, Section 11.7.1, and Section 11.8.8. Note that CQP does not implement a RQ.

The E810 supports a maximum of 256K QPs that are distributed among that active PCI functions using HMC resource profiles. See Section 9.3.3 for more information on the resource distribution mechanism used with QPs.

The E810 maintains the context of each QP in QP Context data structures in the HMC's function private memory space. The storage elements of QP context reside in system memory. See Section 9.3 for more information on the E810's usage of host memory for QP context.

11.4.1.5.1 Doorbell Pages

To support direct access to E810 hardware for PE functionality, the E810 implements adapter memory that is mappable to a userspace process. The memory is exposed to the system through the PCI BAR registers in a similar manner to the CSRs. Each PCI function optionally has a number of PE pages as shown in Figure 11-11.

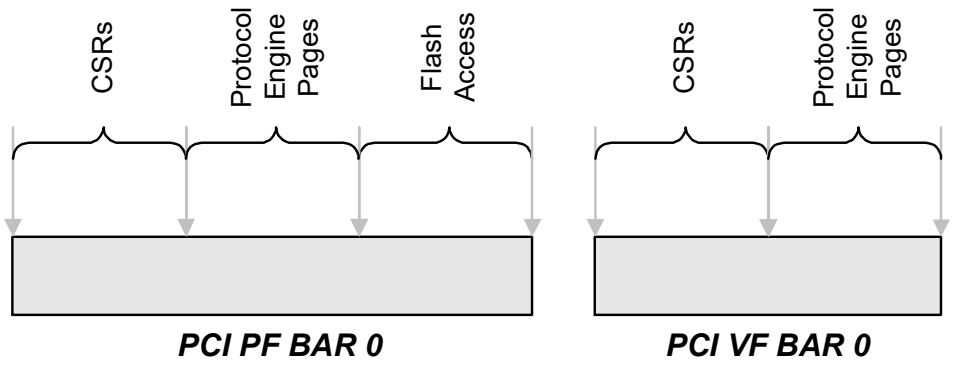


Figure 11-11. PE Page Location in BARs

As shown in Figure 11-12, the first PE page is used for submitting work to PE SQs and also for arming CQs from userspace. 60 KB of space is reserved and then the remaining pages are used to support low latency push mode operation that is described further in Section 11.4.1.5.6.

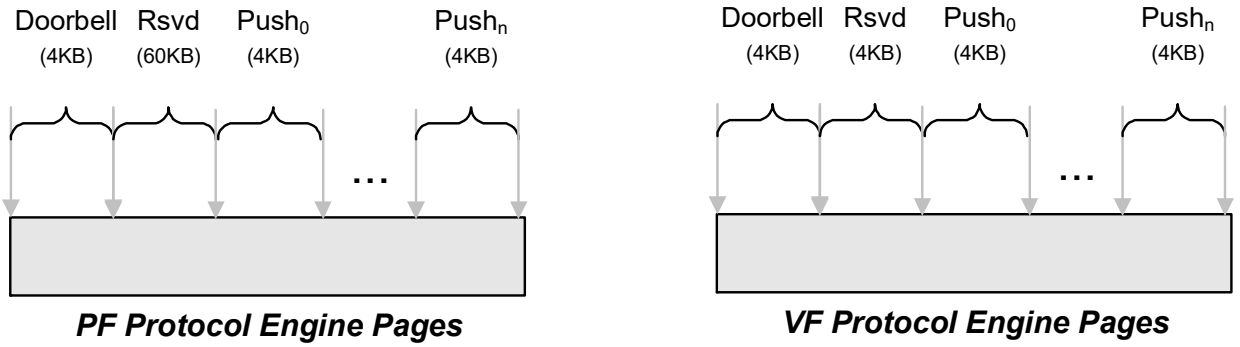


Figure 11-12. PE Pages

The format of the doorbell portion of the PE pages is listed in Table 11-2. The details of each register found in the PE pages are documented in Section 13, "Programming Interface".

Table 11-2. Doorbell Page Register Summary

Offset/Alias Offset	Abbreviation	Name	Block	RW	Section Reference
0x00000000	PFPE_WQEALLOC	PE WQE Allocate Register	PE	RW	13.2.2.28.19
0x00000040	PFPE_CQARM	PE CQ Arm	PE	RW	13.2.2.28.8

11.4.1.5.2 SQ

As shown in Figure 11-13, SQs are organized as circular array of SQ Elements (SQEs). The host memory used for SQs might be physically contiguous or dis-contiguous. Each SQE is 32 bytes in size and can have up to 13 additional fragments for a total of 14. This approach enables Work Requests (WRs) to consume a variable amount of SQ space that enables software to advertise 14 fragments as the maximum supported, but opportunistically pack multiple WRs into a smaller space if the application needed fewer fragments for a given WR.

The E810 takes advantage of variable size WRs by reading 256 bytes at a time from the SQ when processing work. If a WR used all 14 fragments, it would consume the 256 bytes and would be transmitted by itself. If instead the application happened to be using eight consecutive single fragment WRs, all eight WRs would be fetched (assuming software posted the WRs before the E810 managed to read the first WR) with a single read of the SQ, which boosts bus efficiency.

The minimum size of a SQ is four maximum-sized WQEs for proper operations. For example, this means that if a WR can ever use 14 fragments, the SQ must be 1024 bytes. An additional optimization (iWARP only) that the E810 provides is that WRs that are small enough to fit into a single Ethernet packet and fetched as part of the same read of the SQ are sent as a single Ethernet packet for efficiency on the Ethernet fabric.

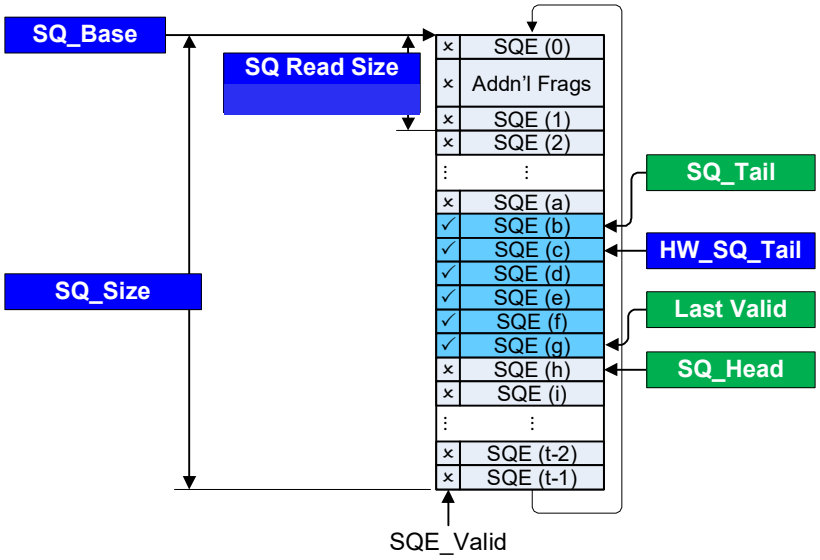


Figure 11-13. PE SQ

Figure 11-13 depicts a typical SQ, where software is tracking *SQ_Head* and *SQ_Tail*, and the E810 tracks *HW_SQ_tail* in QP context. Software must size the SQ large enough to support every SQE with the maximum number of additional fragments. The amount of space required per work request is listed in Table 11-3. Essentially the WR space requirements on the SQ end up rounding up to the next multiple of 32 bytes. SQE 0 is shown using four additional fragments and therefor takes up 96 bytes.

Every SQE must start on an offset that is a multiple of 32 bytes within the SQ and an SQE, and all of related additional fragments must fit into a single 256 byte SQ read block. NOP SQEs can be added if necessary to pad SQEs out when necessary to satisfy this requirement. The 32-byte boundary requirements for SQEs means that there is no difference in SQ size between a WR with four fragments and a WR with five fragments. If SQE 1 required any additional fragments, a single NOP SQE would need to be added in place of SQE 1 and all subsequent SQEs would move down in the SQ so that the SQE 1 and all related additional fragments would fit in the same 256 byte SQ read block.

For the SQ, the maximum size of inline data must also be taken into account when calculating the amount of space required per work request (see Table 11-3).

Table 11-3. QP WR Size Requirements

Number of Fragments Requested at QP Creation	Max Inline Size (SQ Only)	Size Required per Work Request
1	≤ 8 bytes	32 bytes (1 SQE)
2-3	≤ 39 bytes	64 bytes (1 SQE + 2 additional fragments)
4-5 (SQ only)	≤ 70 bytes	96 bytes (1 SQE + 4 additional fragments)
6-7	≤ 101 bytes	128 bytes (1 SQE + 6 additional fragments)
8-9	≤ 132 bytes	160 bytes (1 SQE + 8 additional fragments)
10-11	≤ 163 bytes	192 bytes (1 SQE + 10 additional fragments)
12-13	≤ 194 bytes	224 bytes (1 SQE + 12 additional fragments)
14	≤ 224 bytes	256 bytes (1 SQE + 13 additional fragments)

Figure 11-14 shows a system view of how the E810 processes PE QP SQs. Before describing the flow in Figure 11-14, software must perform a fair amount of initialization. Software starts by populating all QP-related HMC objects necessary for CQs, QPs as well as related memory regions. Next, software allocates buffers for software QP and CQ context, SQ and RQ WRID tracking arrays (software context and WRID tracking arrays do not need to be pinned, QP40, CQ48, and CQ41 in Figure 11-14), QP and CQ doorbell shadow/status areas, Q2 areas, and finally the SQ, RQ, and CQ buffers themselves.

The doorbell shadow/status areas, Q2 and WQs all need to be pinned, and if they are physically dis-contiguous, the page list for each of the pinned buffers must be collected. CQP operations are used to create the CQs (if they do not already exist) and QPs.

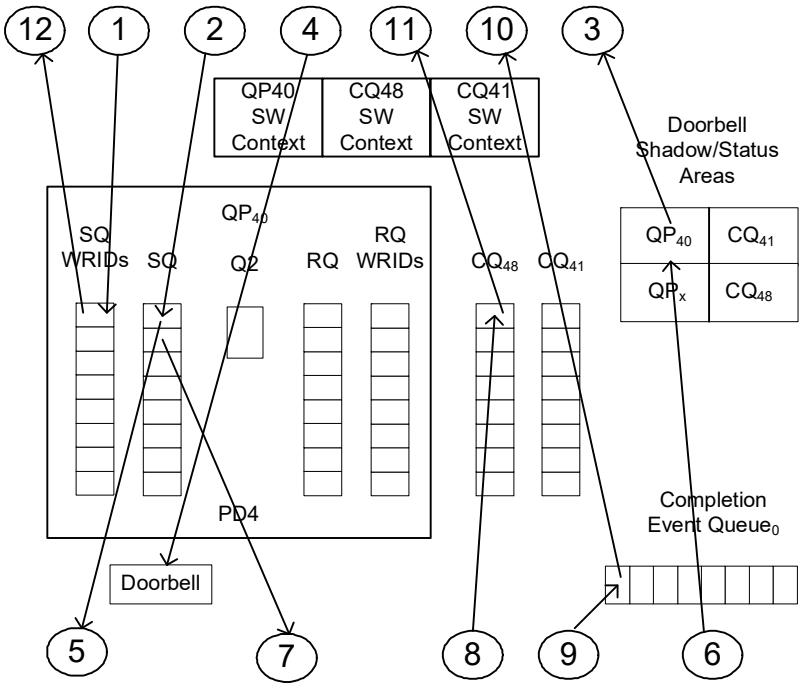


Figure 11-14. PE Operation: SQ

Now that all structures have been created and defined to the E810, the steps shown in Figure 11-14 to process a post-send verbs request are the following:

1. Software stores the WRID supplied by the application on the post-send request to the SQ WRID.
2. Software creates a SQE along with any additional fragments necessary to represent the WR.
   a. If the SQE being created does not consume the rest of the 256-byte SQ read block, software must ensure that the next SQE has the *Valid* bit set to the invalid setting. This is not required if the next SQE starts in a different 256-byte SQ read block.
   b. Step 2a must be done before this step. The *Valid* bit in the SQE must only be set once after all fields are valid, since the E810 can read SQEs at any time. Note that the *Valid* bit is a generational valid bit, which means that for the first (and all subsequent odd) iteration through the SQ, 1b means valid. On all even iterations through the SQ, 0b indicates that the SQE is valid.
3. Software reads *HW_SQ_Tail* from the doorbell status area for the QP.

4. Software determines whether or not to ring the SQ doorbell.
    a. If *HW_SQ_Tail* is in the range of WQEs just posted (taking into account the possibility of wrapping), ring the SQ doorbell.
    b. *SQE_Head* is incremented by the number of SQEs posted since the last doorbell check (WR size in bytes/32 bytes).
5. The E810 reads the 256-byte SQ read block that contains the SQE indicated by QP context value of *HW_SQ_Tail*.
    a. The E810 processes all valid SQEs contained in the 256 byte SQ read block. If no SQEs are valid, proceed to Step 6.
    b. The E810 processes the SQEs, which likely includes fetching data from host memory, sending packets through the Ethernet port, and waiting for TCP acknowledgments.
    c. The E810 can increment the *HW_SQ_Tail* in the doorbell shadow/status area during this processing.
    d. If the last SQE in the 256-byte SQ read block is valid, Step 5 is repeated, which reads the next 256-byte SQ read bock.
6. The E810 writes the doorbell status area for the QP with the current value of *HW_SQ_Tail* or the write can be delayed until later.
7. The E810 reads the 256-byte SQ read block one more time to ensure that software has not posted new work.
    a. If a new valid SQE is found, proceed back to Step 5a.
8. The E810 optionally writes to the CQ associated with the SQ.
9. The E810 optionally writes to the CEQ if the CQ was written and generates an interrupt.
10. Software then fields the interrupt if one was generated and reads the CEQ to determine the CQ that generated the event (an application might be polling the CQ directly in which case Step 9 and Step 10 are skipped). Typically the CEQE contains the most significant 63 bits of the virtual address of CQ context to make CE processing efficient.
11. Software then reads the CQ to determine the QP and WQ that generated the work completion. The SQE index of the WR is reported in the CQE along with a 64-bit pointer that is typically set to the Software QP context address at QP creation time.
12. Software then reads the SQ WRID array for the SQE index from the completion. The resulting WRID is returned to the application as an indication that the original work request completed.

Note: While Step 6 and Step 8 are shown as sequential, the E810 starts both steps simultaneously. Also, Step 1 can be initiated by software at any time as well.

11.4.1.5.2.1 Immediate Data Operations

The E810 supports Immediate Data operation for the SQ.

In iWARP, the intention of this operation was to combine it with a previously-adjacent RDMA Write operation to emulate an RDMA Write with Immediate Data operation. iWARP supports a Write with Immediate Data operation. Send with Immediate Data is not defined for iWARP and is not supported. From a wire perspective, the Immediate data is an untagged operation and consumes an RQ WQE and also a CQE at the peer.

RoCEv2 supports Send with Immediate Data and Write with Immediate Data.

11.4.1.5.2.2 RDMA Read with Multiple SGEs or Local Data Sink Buffer

The InfiniBand definition for RDMA Read has more flexibility than iWARP:

- InfiniBand supports multiple SGEs. iWARP does not.
- InfiniBand does not require remote write access on sink buffers. iWARP does.
- InfiniBand allows privileged applications to use the Reserved L-Key in the SGEs. Because of the remote write access requirement, iWARP does not allow STag 0.

The last item especially puts iWARP at a disadvantage since it requires privileged applications to do fast register and invalidates around RDMA Reads.

The E810 supports RoCEv2, so it has to implement the more general RDMA Read semantics. The E810 provides a way to eliminate these differences for iWARP without changing the wire protocol.

In general, this approach enables multiple data sink SGEs to be used for RDMA Reads by using the *AdditionalFragmentCount* field of the RDMA Read WQE. When the Read Response comes in, the hardware re-read the WQE to determine where to place the data. The hardware validates that the data can be placed against each SGE.

11.4.1.5.2.2.1 iWARP - RDMA Read with Multiple SGEs

Hardware generates an artificial data sink STag. The low 14 bits have the index to the RDMA Read WQE in the SQ. The high bits are randomized and are guaranteed to be non-zero.

When the Read Response comes in, the WQE index in the STag field is used to tell hardware where the WQE is in the SQ. Hardware re-reads the WQE to determine where to place the data. Standard STag validation is done on the RDMA Read WQE SGEs. The exception for the WQE SGEs is that Remote Write access is not required. Since the STag in the Read Response directly references the SQ WQE, there is no additional object required.

The artificial STag is very specific to RDMA Reads and is not used for any other purpose. Nor is it known by software. It is possible that a real STag and an artificial STag could use the same value. Hardware will not have problems since the two values are used in different ways.

Note: The *Infiniband_read_en* flag must be set in the QP context to support InfiniBand style RDMA reads for iWARP. If this flag is not set, iWARP RDMA Reads operate the old way (one SGE, *write_access* required, no support for STag 0).

11.4.1.5.2.2.2 RoCEv2 - RDMA Read

In RoCEv2, hardware allocates a Read Response Entry object before the RDMA Read Request is sent. This object tracks the index of the WQE that needs to be re-read. When a Read Response is received, the SQ WQE index is used to re-read the WQE. When the RDMA Read completes, hardware places the object on the Read Response Entry Free List.

11.4.1.5.3 Memory Keys

When iWARP registers a block of memory, it allocates an STag, which is used for local and remote operations. RoCEv2 defines separate keys for local operations (L_Key) and remote operations (R_Key). The E810 generates one key for registered memory. The key is used for both the L_Key and R_Key values.

This document uses the term STag generically when referring to memory keys.

11.4.1.5.4 Privileged Keys

The use of a privileged key allows QPs to reference physical addresses. They are never placed on the wire, and only privileged QPs are allowed to use them. For this purpose, iWARP defines STag 0, and RoCEv2 defines a Reserved L_Key.

Unlike iWARP's STag 0, RoCEv2 does not architecturally define the value for the Reserved L_Key - it is defined by the device. This implementation uses the value of 0 for the Reserved L_Key.

11.4.1.5.5 Virtual Queues

The E810 supports physically contiguous and physically dis-contiguous WQs for PE QPs. CQs, and CEQs also support both physically contiguous and dis-contiguous buffers. In the case of QPs, the support for physically dis-contiguous WQs is referred to a virtual WQ support. This section describes an example of a SQ.

Figure 11-15 shows a virtually contiguous but physically dis-contiguous SQ. The left side of the figure shows the applications view of the SQ. This particular SQ is composed of four host memory physical pages that are not contiguous. In this scenario, the virtual buffer is pinned and the 1 level page list is retrieved from the operating system. Then an unused page list must be allocated from the HMC PBLE object space with four contiguous PBLEs. This PBL is now populated with the page list retrieved from the operating system. When the QP is created, the *Virtual_WQs* bit is set in CQP Create QP WQE and the *SQ_Base* in QP context is set to the starting PBLE HMC object index instead of a host physical address of the SQ. The RQ for a QP with the *Virtual_WQs* bit set must also use a PBL even if it happens to be physically contiguous. Note that the SQ and RQ in this case must be allocated a host page boundary.

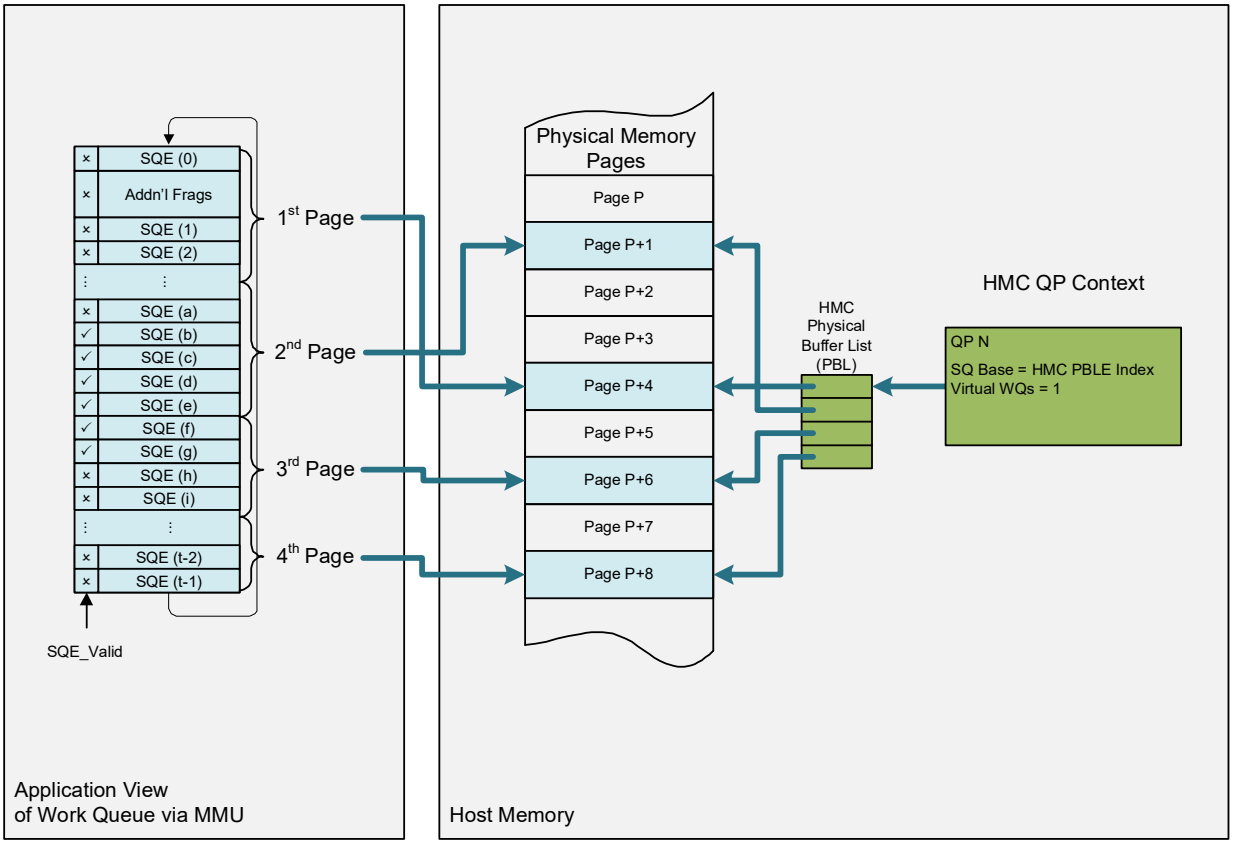


Figure 11-15. PE Operation: Virtual SQ

11.4.1.5.6 SQ Push Mode

Push mode reduces processing latency of the short message posted for transmission on the low-latency QP by eliminating a read of WQE or WQE with inline data from the host memory. This is achieved by software writing WQEs to the adapter memory-mapped address space using processor write-combining buffers. Processor write-combined buffers allow combine writes and make transactions on the bus more efficient. Software is not expected to use this mode to stream high rate of the short messages or use this mode to post a large messages. The size of the message posted using push mode should not exceed a configured scheduling quanta. This feature can be either exposed to the application and have an application responsible for the potential QP bandwidth degradation caused by overuse of this feature, or a verb layer can be instrumented to use information provided by hardware and opportunistically take advantage of push mode messages for the given connection based on true low-latency messaging requirements.

In modern systems, a memory type can be specified within an operating system page structure using an index in the PAT register. This way an individual page of adapter memory can be mapped to the application address space as a write-combined memory. Writes to such page would be combined in the processor write combining buffers. These buffers are flushed on PCI bus in cache lines. In some systems, the same page can be mapped as an un-cached memory page that enables software to use un-cached write (or doorbell ring) to flush push mode WQEs from processor write-combining buffers to the PCIe bus.

Adapter memory-mapped pages used for push mode WQEs are called push pages. The number of push pages exposed to the driver depends on the chip configuration and PCIe function. In most systems, it is not possible to map the same page as both un-cached (non-write-combined) and write-combined. In this case, the GLPE_PSHCFG.*PSHCFG_DB_SPLIT* bit must be set, which forces the doorbell pages to be treated separately from the page used to push the WQEs and data. When GLPE_PSHCFG.*PSHCFG_DB_SPLIT* is set, two push pages are mapped to the application address space where the odd number push page is mapped as un-cached (used as the push doorbell page) and the even numbered push page is mapped as write-combined. This reduces the amount of push pages available to applications by half since a single application requires two separate push pages for proper operation.

Combining writes using dedicated write-combining buffers is a very old feature of the processors. This feature was mainly used for the ancient graphic adapters. Now days all graphics adapters use their own DMA engines to pull the data from the host memory, rather than having it pushed with CPU.

Processors specify a write combining memory region using several mechanisms.

- The MTRR register specifies a memory type per region. Granularity of memory region sizes does not allow the use of MTRR for wide deployment of write-combining for the E810.
- Page Attribute Table (PAT) in conjunction with flags in Page Table Entry (PTE) enables write-combining on a per-page base, which makes it feasible for E810 deployment. Relatively recent distribution of Linux added an interface allowing change page attributes to write-combined page.

Processors have a set of buffers for write-combined accesses. Each buffer has a size of several cache lines. Writes to write-combined memory are intended to be delayed to enable better memory access efficiency and reduced overhead.

Write-combined buffers are evicted or flushed by serialization events, such as SFENCE, LOCK, interrupt, and so on, and read/write to un-cached memory. Partial write-combined buffer eviction can be a result of the serialization event. Eviction of a full write-combined buffer is implementation dependent. Intel assumes that serialization events are evicting/flushing both full and partially filled write-combined buffers. The intention is to use doorbell ring (write to un-cached memory) as a serialization event to evict write-combined buffers.

All types of the PE SQ WQEs can posted using a push mechanism. The maximum size of data that can be pushed together with WQE as an inline data is limited to the 224 bytes, matching the maximum size of data that can be posted using WQE with inline data. Larger messages can be pushed using regular WQE, eliminating a need to read a WQE from the host memory. If an application requested post send, verb layer can decide whether to push the message to adapter based on the message size, configurable threshold and outstanding WQEs previously posted to SQ. In any case, verb layer should always first post WQE to the SQ (without ringing the doorbell) and only then write WQE push mode page, followed by un-cached write to the associated push mode page. Posting WQE to the SQ is required for the proper re-transmission process, and enables hardware to opportunistically discard push mode messages and transmit those using regular SQ processing mechanism.

To avoid security implications, a push mode page cannot be shared by different processes. If it was allowed to be shared, then ProcessA could guess the QP number of ProcessB and other QP attributes, like the push mode page, and make ProcessB send a message pushed by ProcessA as its own. This can lead to a severe security violation. Multiple threads residing in the same process share a process virtual space and belong to the same security domain. The number of push pages for VFs is limited to 15 to reduce the amount of BAR space that the operating system is required to allocate for the E810. To overcome this restriction, a para-virtualized driver is allowed to map PF push pages to the VF for use.

Push mode is an opportunistic latency optimization that is supposed to be used on lightly loaded QPs. If the adapter runs out of resources or software attempts to use push mode WQEs too aggressively, the adapter can discard a push mode WQE, and transmit it later as a regular WQE with inline data posted to SQ.

Software is allowed to post any SQ message using push mechanism. The size of the message should not exceed a configurable quanta value (default of 4 KB). Messages specified as inline operations by application can be pushed together with data. A total of 224 bytes of the message with inline data can be pushed to the E810 to avoid overhead of the SQE and data fetches. Larger messages would avoid a fetch of the SQE element, but still requires a data fetch. Software can post multiple messages with single push operation, as long as all messages are continuously located and do not cross a boundary of the push page SQE element, shown in Figure 11-16 and Figure 11-17.

Push mode does not require any change in WQE format, except for setting a push bit in the SQ WQE. A process is enabled for push mode by allocating a push page shown in Figure 11-16, or a pair of push pages as shown in Figure 11-17 (see Section 11.5.3.8 for the CQP WQE format) from the E810's BAR to a QP and including the push page Index in QP context when the QP is created. Each PCI function has a number of Push mode pages that must be assigned to a queue set handle. See Figure 11-11 and Figure 11-12 for more details on how the E810 exposes push pages to the system. The E810 validates that the push operation is associated with the correct push page and Push SQE index as well as PCI function before acting on the operation.

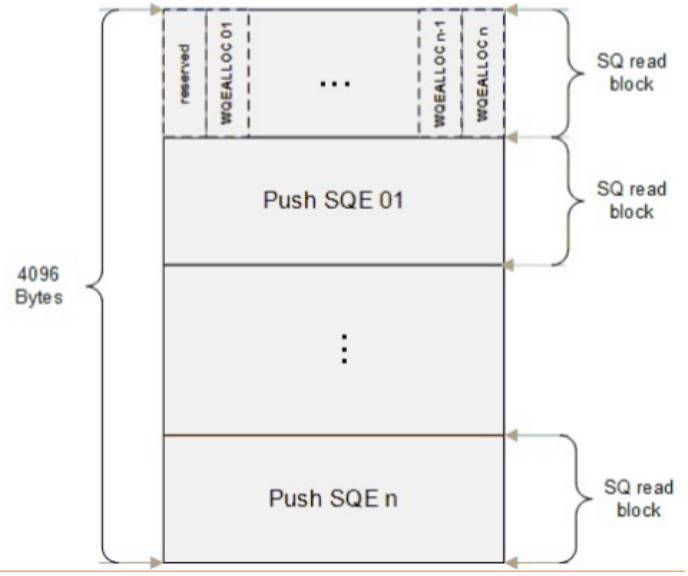


Figure 11-16. Combined Push Page Detail

On a combined push page, the number of WQEs is 15.

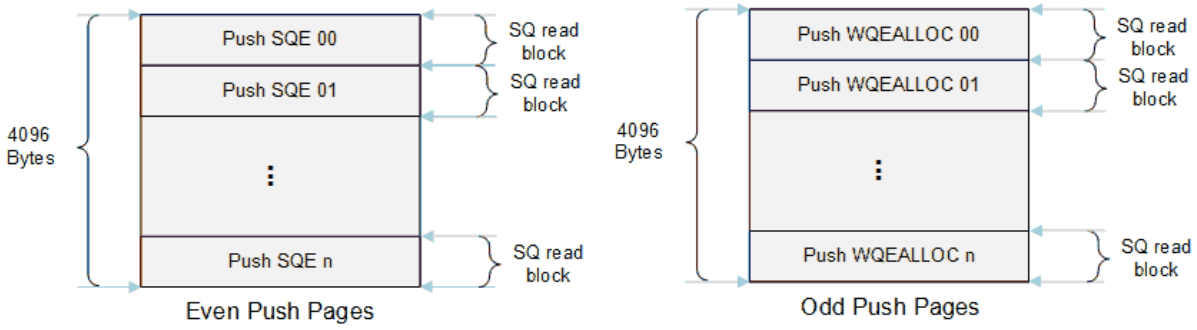


Figure 11-17. Split Push Page Detail

On a split push page, the number of WQEs is 16.

When GLPE_PSHCFG.*PSHCFG_DB_SPLIT* is clear, the push doorbell area is located at first 256-byte region of the push page. Push doorbell area for PFs contains 31 instances of the PFPE_WQEALLOC register described in Section 13.2.2.28.19. PFPE_WQEALLOC instances are continuously located within the push doorbell area starting at offset four bytes into the 256-byte region. Push doorbell area for VFs contains 31 instances of the VFPE_WQEALLOC register described in Section 13.2.2.28.18. VFPE_WQEALLOC instances are continuously located within the Push Doorbell area starting at offset four bytes into the 256-byte region. All QPs associated with a given push page use the same push doorbell area. Each instance of the PFPE_WQEALLOC and VFPE_WQEALLOC corresponds to the respective Push SQE. The Push SQE index matches an index of the respective PFPE_WQEALLOC or VFPE_WQEALLOC registers.

When GLPE_PSHCFG.*PSHCFG_DB_SPLIT* is set, the WQEALLOC instances are located in odd numbered push pages and the PUSH SQEs are located in even numbered push pages. Additionally, the stride of the doorbell array to 256 bytes instead of the packed array of doorbells shown in Figure 11-16. In other words, each odd numbered push page has WQEALLOC instances at offsets 0, 256, 512, ... instead of 0, 4, 8, ... to allow for write-combine avoidance on un-cached writes to the push doorbells.

Figure 11-18 shows the mechanism for pushing a WQE to the E810.

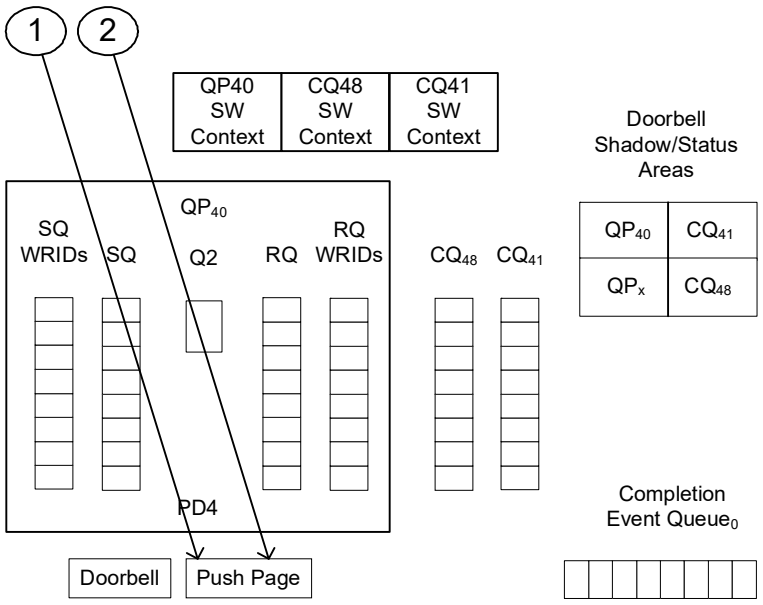


Figure 11-18. PE Operation: SQ Push Mode

Push mode operation builds on the typical SQ operation shown in Figure 11-14. The following two steps replace Step 4 from the procedure defined in Section 11.4.1.5.2.

For the Push operation to be effective, the SQ should be empty before placing WQEs. Software can place multiple Push WQEs (that is, the SQ does not have to be empty as long as they are all Push WQEs). Once a non-Push WQE is placed to the SQ, software should not place any more Push WQEs until the SQ is empty again. In other words, software should not intermix Push and non-Push WQEs. Additionally, if a CQE indicates that a Push WQE has not completed as a Push WQE, software should cease posting Push WQEs until the SQ is empty again.

1. Instead of the usual doorbell page write, a copy of the SQE is written to one of the push page SQEs from Figure 11-16. This page is configured with write combine attributes for this step. If less than 256 bytes WQEs are written to the 256-byte Push WQE and the WQE did not start on a 256-byte boundary within the SQ, the WQE data from the SQ must be written to the matching offset in the Push WQE from the start of the most recent 256-byte boundary in the SQ. For example, if WQE index 9 from the SQ happens to be 32 bytes long and needs to be pushed, 32 bytes need to be written 32 bytes past the start of the Push WQE instead of at offset 0 in the Push WQE. This is because WQE index 9 starts 32 bytes past the previous 256-byte boundary in the SQ. Write to the Push WQE must be eight bytes at a time, on 8-byte boundaries, and all bytes of a Push WQE must be written. There must be no partial cache line writes. If writes are not on natural 8-byte boundaries, the Push WQE is dropped.
2. After writing the SQE to the push page SQE, the push page doorbell is written to indicate that the push is complete. Note that the doorbell written for non-push operations is not written for push operations.

Step 7 from the procedure described in Section 11.4.1.5.2 is typically skipped for push mode operations unless the E810 is low on resources, the traffic class associated with the push page has insufficient bandwidth allocation to complete the operation. Push mode operations are also ignored if the traffic class associated with the push page is flow controlled. In the cases where the E810 ignores the SQE pushed in Step 2 above, the WQE is re-fetched normally and processing continues. The only side effects of the dropped push operation is wasted bandwidth on the PCI bus and wasted CPU cycles. To keep the wasted bandwidth and CPU cycles to a minimum, the next CQE generated after the dropped push operation indicates that the drop occurred and software should stop issuing push operations until it reaches an empty SQ condition.

The maximum number of pending push doorbell resources is 128. Software can prepare the next 128 WQEs, but it should not exceed 128 pending push doorbells. If software exceeds this amount, the hardware directs the push doorbell to the SQ.

11.4.1.5.7 SQ Suspend and Resume

Software is provided with CQP Commands allowing a suspend and resume operation of the SQ. Commands are defined in Section 11.5.3.22 and Section 11.5.3.23, respectively. Those commands are primarily used to migrate QP from one QS to another QS. This migration might be triggered by a system configuration change. For example, a change in number of TCs configured for the particular physical port, which in turn requires reassignment of QPs to the QSs associated with the TCs.

Software is required to reassign all QPs associated with particular QS prior posting an AQ Command that might result in QS removal. Such reassignment of the QP to the new QS should be performed using Suspend and Resume CQP commands (Section 11.5.3.22 and Section 11.5.3.23, respectively). Both Suspend and Resume CQP commands are asynchronous requests, and their completion does not indicate completion of a requested operation. Completion of a resume operation is never reported to software. Completion of a suspend operation is reported using AE_QP_SUSPEND_COMPLETE. Software is allowed to request multiple suspend and resume operations for different QPs accelerating transfer of multiple QPs from one QS to another.

11.4.1.5.8 RQ

As shown in Figure 11-13, RQs are organized as circular array of RQ Elements (RQEs). The host memory used for RQs can be physically contiguous or dis-contiguous. Each RQE is 32 bytes in size and can have up to 13 additional fragments for a total of 14. Unlike SQs, RQs all have fixed size Work Requests (WR) even if the application uses a smaller number of fragments that could be used to represent a WR. The minimum size of a RQ is four maximum sized WQEs for proper operations. For example, this means that if a WR can ever use 14 fragments, then the RQ must be 1024 bytes. This is necessary to allow out-of-order placement of inbound RDMA send operations for iWARP QPs. All RQEs must start on an offset that is a multiple of RQ WQE size and an RQE along with its associated additional fragments must be contained in a single 256-byte RQ read size buffer. This leads to the same RQ buffering requirements per WR as those listed for the SQ in Table 11-3.

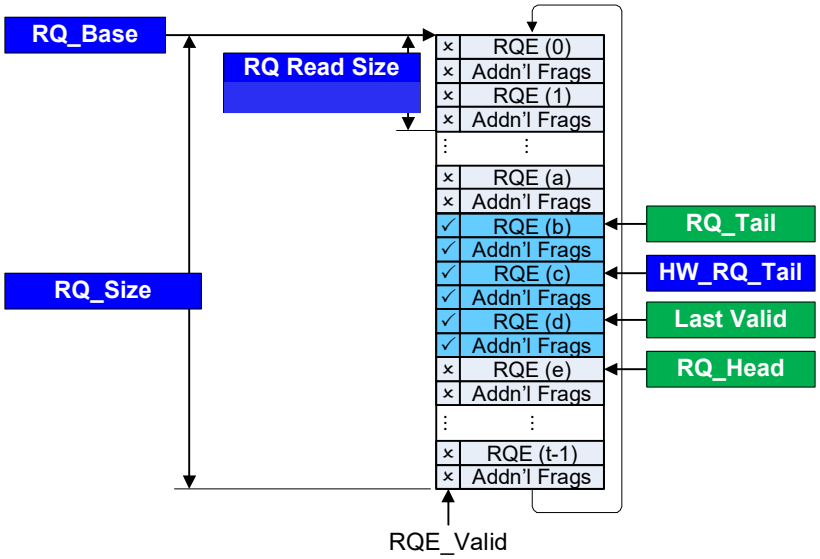


Figure 11-19. PE RQ

Figure 11-19 shows a typical RQ where software is tracking *RQ_Head* and *RQ_Tail*, and the E810 tracks *HW_RQ_Tail* in QP context. Software must size the RQ large enough to support every RQE with the maximum number of additional fragments. All RQEs are shown using two additional fragments and therefore takes 64 bytes. NOP RQEs can be added if necessary to pad RQEs out when necessary to satisfy the RQE alignment requirements. The 32-byte boundary requirements for RQEs means that there is no difference in RQE size between a WR with two fragments and a WR with three fragments.

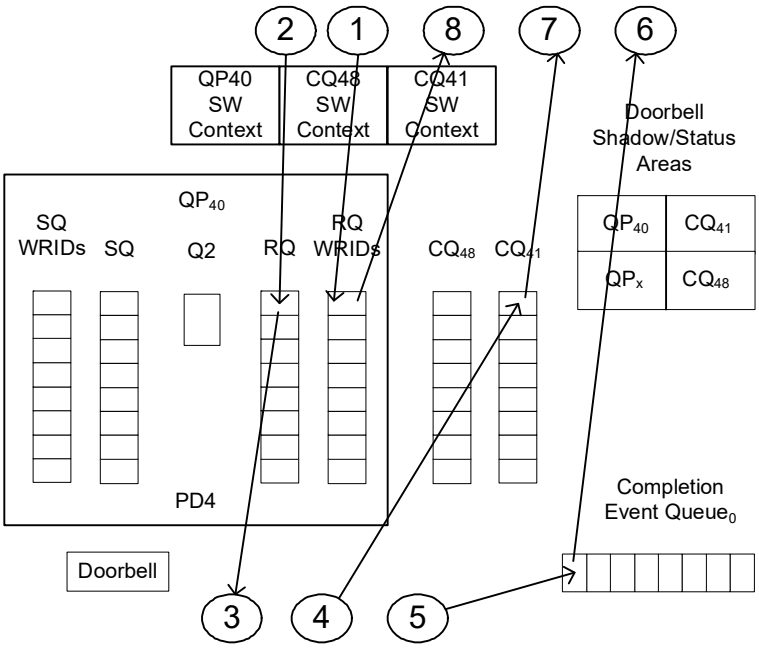


Figure 11-20. PE Operation: RQ

The initialization required to bring a QP to an operation state was generally described in Section 11.4.1.5.2. Once this initialization has been performed, the steps in Figure 11-20 show how software and the E810 interact for RQ operations.

1. Software saves the Work Request ID (WRID) passed in on the post-receive verbs call.
2. Software builds a RQE using the fields that passed in on the post-receive verbs call.
   a. The *Valid* bit in the RQE must only be set once all fields are valid since the E810 might read RQEs at any time. Note that the *Valid* bit is a generational valid bit, which means that the first (and all subsequent odd) iterations through the RQ, 1 means valid, on all even iterations through the RQ, 0 indicates that the RQE is valid.
3. Once an operation targeting a QPs RQ is received, the E810 reads QP context to determine the location of the RQE to be read.
   a. The RQE is read from host memory and the *Valid* bit is checked. If the RQE is invalid, the packet associated with RQE is dropped, or the QP is put into the error state and an AE is generated.
4. The E810 reads the 256-byte RQ read block that contains the RQE indicated by the QP context value of *HW_RQ_Tail*.
   a. The E810 potentially processes all valid RQEs contained in the 256-byte RQ read block if more than one inbound operation targeted this QP's RQ. For this example, it is assumed that a single inbound operation has been received.
   b. The E810 processes RQE 0, which likely includes writing data to host memory and sending TCP acknowledgments.
   c. The E810 increments the *HW_RQ_Tail*.
5. The E810 writes to the CQ associated with the RQ.
6. The E810 optionally writes to the CEQ if the CQ was written and generates an interrupt.

7. Software then fields the interrupt if one was generated and reads the CEQ to determine the CQ that generated the event (an application might be polling the CQ direction in which case Step 6 and Step 7 are skipped). Typically, the CEQE contains the most significant 63 bits of the virtual address of CQ context to make CE processing efficient.
8. Software then reads the CQ to determine the QP and WQ that generated the work completion. The RQE index of the WR is reported in the CQE along with a 64-bit pointer that is typically set to the software QP context address at QP creation time.
9. Software then reads the RQ WRID array for the RQE index from the completion. The resulting WRID is returned to the application as an indication that the original WR has completed.

11.4.2 iWARP State Management

iWARP QP state is controlled by the application to a large extent. The verbs interface enables the application to manage QP transitions and provides rules for what the RNIC Interface (RI) must enforce.

This section is not intended to repeat all information from the verbs specification. It is intended to be an overview of the expected usage model to be used with the E810. In the end, it is a combination of hardware support from the E810 and supporting software that provides a verbs-compliant interface.

Table 11-4 lists the operations performed by the E810 based on the iWARP QP state.

Table 11-4. iWARP QP State Behavior

QP State	QP Activities
Non-existent	• Doorbell rings are dropped silently. • No transmit (because doorbell rings are dropped). • No receive processing (no hash entry).
Idle	• SQ/RQ WQEs can be posted. • No SQ WQEs are processed. • No receive processing (no hash entry).
RTS	• Normal SQ/RQ/CQ/CEQ processing (hash entry added on transition to RTS by firmware).
Closing	• No new SQ work is expected. • Doorbell rings are dropped silently. • No new RQ work is expected. However, if a packet arrives with data it is treated as an error (BAD_CLOSE) and the QP transitions to error state.
Terminate	• No RQ processing (discarding received data). • No new SQ WQEs processed. • SQ WQEs might be retransmitted. • Terminate/FIN might be transmitted.
Error	• No SQ WQEs processed. • No RQ WQEs processed. • Host software eventually requests CQP to remove the hash entry.

QP state transitions and the associated software and hardware/firmware actions driven from software interactions with the application are listed in Table 11-5.

Table 11-5. iWARP QP State Transition

Initial and Next QP State	Software Actions During State Transition	Hardware Actions During State Transition
Nonexistent -> Idle (CreateQP)	• Initialize the HMC backing pages necessary to support the QP. • Allocate/Initialize all software tracking structures for the PD, CQ, QP. • Assign QP to a PD. • Create CQ. • Allocate SQ and RQ. • Allocate terminate message buffer (Q2). • Allocate the doorbell shadow area. • Perform parameter checks specified by the verbs specification. • Issue CreateQP WQE to CQP.	• Update QP context.
Idle -> Idle	• Perform parameter checks specified by the verbs specification. • Issue Modify QP to CQP, next state = Idle.	• QP state parameters are updated.
Idle -> RTS	• Create a TCP socket and ensure that it is in the established state. • If a streaming mode message is desired, ensure that it has been placed on the SQ immediately following any unacknowledged TCP send data. • Perform parameter checks specified by the verbs specification. • Issue Modify QP to CQP, next state = RTS.	• Set the TCP state to established and RDMA state to RTS. • Insert the QP's hash table entry to receive accelerated data. • Start scheduling work for the QP.
Idle -> Error	• Perform parameter checks specified by the verbs specification. • Issue Modify QP WQE to CQP with the next state set to error, reset connection (if connection not already reset) and remove quad hash (if quad hash entry has not been previously removed) by setting appropriate bits in the bits set. • If any WQEs are still pending, issue flush WQEs operation to CQP after the Modify QP operation completes, if the QP has been setup with a last streaming mode message, the first WQE on the SQ is the streaming mode message, not an iWARP message, a unique completion context should be put in the LSSM WQE in order to correctly distinguish it from iWARP WQEs. • Handle poll for completion requests for flushed WQEs, the CQ has one CQE for each WQ that had pending WQEs, software must report all pending WQEs as flushed once it gets the single flushed CQE (per WQ with pending WQEs), the number of WQEs flushed can be determined from software's head and tail for the SQ and RQ at the point in time that the poll for completion call is made that returns a flushed CQE.	• Set RDMA state to error. • When the flush WQEs operation is issued and any WQEs are outstanding on the SQ and/or RQ as requested by host software, complete the first pending WQE for each WQ (flushed completion status).

Table 11-5. iWARP QP State Transition [continued]

Initial and Next QP State	Software Actions During State Transition	Hardware Actions During State Transition
RTS -> Closing	• Perform parameter checks specified by the verbs specification. • Stop processing new PostSQ/RQ requests. • Ensure that no outstanding SQ or outbound RDMA read responses are pending (this is really up to the application). • Start a timer to put a bound on how long a connection is allowed to stay in the closing state. • Submit a Modify QP WQE to CQP the next state = Closing. • If the completion error is a bad close, generate an AE, QP state is already error. • If any RQ WQEs are still pending, issue Flush WQEs operation to CQP.	• If there is pending SQ or Q1 work, generate a completion error (bad close), software needs to translate into an AE. • Else, update RDMA state to closing. • When the flush WQEs operation is issued and any RQ WQEs are outstanding, complete the first pending WQE for each WQ (flushed completion status).
RTS -> RTS	• Perform parameter checks specified by the verbs specification. • Ensure that the number of currently pending outbound RDMA read requests is less than or equal to the new ORD value. • Ensure the only parameter that is changed is ORD size, it is only allowed to shrink or remain the same.	• Update the ORD value.
RTS -> Terminate or Terminate -> Terminate	• Perform parameter checks specified by the verbs specification for the RTS -> Terminate case. • Set terminate action (usually send FIN and terminate). • Start a timer to put a bound on how long the QP is allowed to stay in the terminate state in the case of an abortive tear-down. • If completion error indicates tcp_state = error: — Issue Modify QP WQE to CQP with next state = Error. — Generate LLP connection reset AE. • If terminate sent AE is received, issue an additional Modify QP with next state = Terminate and terminate action set to 3 (do not sent FIN or terminate). • If completion code indicates LLP closed, the TCP state has transitioned to closed or time wait so an LLP close complete AE should be generated by software.	• If the TCP state = Closed, generate completion error indicating *tcp_state* was closed (indicates a reset was received). • Update iWARP state to terminate. • Once all outstanding TCP segments have been acknowledged, send the terminate message and/or FIN. • Generate a terminate sent AE.
RTS -> Error or Closing -> Error or Terminate -> Error	• Perform parameter checks specified by the verbs specification. • See IDLE->Error case for CQP and completion processing. • Block until the RST sent AE has been seen (probably want to start a timer, and upload QP context if it expires to see what has happened). • If any WQEs are still pending, issue flush WQEs operation to CQP after the Modify QP operation completes, if the QP has been setup with a last streaming mode message, the first WQE on the SQ is the streaming mode message, not an iWARP message, a unique completion context should be put in the LSSM WQE in order to correctly distinguish it from iWARP WQEs.	• If host software requested the connection to be reset, send a TCP RST segment. • Remove the QP's HTE if the *Remove_Hash_Entry* field is set. • Update iWARP state to error. • Wait for all packets to clear the pipeline. • Clean up QP context (prevent future scheduling). • Generate a RST sent AE. • Set the TCP state to closed.

Table 11-5. iWARP QP State Transition [continued]

Initial and Next QP State	Software Actions During State Transition	Hardware Actions During State Transition
Error -> Idle	• Software issues destroy QP operation to CQP.	• Destroy the QP.
Error -> Error	• Submit a Modify QP WQE to CQP with next state = Error and the reset_connection bit set. • If any WQEs are still pending, issue flush WQEs operation to CQP after the Modify QP operation completes, if the QP has been setup with a last streaming mode message, the first WQE on the SQ is the streaming mode message, not an iWARP message, a unique completion context should be put in the LSSM WQE in order to correctly distinguish it from iWARP WQEs.	See RTS-> Error.

QP state transitions and the associated software and hardware/firmware actions driven from wire interactions with remote peer are listed in Table 11-6.

Table 11-6. iWARP QP State Transition Driven from the Wire

Row #	Wire Activity	Initial QP State (TCP, iWARP)	E810 Actions	Software Actions
0	FIN Received	EST, RTS	• Set the TCP state to close wait (note that iWARP state is not changed). • If no work was outstanding on the SQ or Q1: — Issue AE_LLP_FIN_RECEIVED. Else: — Issue AE_RDMAP_ROE_BAD_LLP_CLOSE. • Send ACK to the FIN. • Continue processing SQ WQEs.	• Stop processing new post RQ operations if not already stopped. • Issue appropriate AEs. • If system software does not automatically generate closing, start the closing process based on the received event by issuing Modify QP with the next state set to closing.
1	FIN Received	FW1, Closing	• Send ACK to the FIN. • If the ACK receive with the inbound FIN acks the transmitted FIN: — Set the TCP state to Time Wait. — Issue AE_LLP_CLOSE_COMPLETE. Else: — Set the TCP state to Closing (note that iWARP state is not changed).	• If the AE_LLP_CLOSE_COMPLETE is received: — Cancel the pending disconnect fail safe timer. — Stop processing new post RQ operations if not already stopped. — SQ operation processing should have been stopped on the Modify QP with the next state set to closing. — Issue appropriate AEs. — Issue Flush WQE CQP operation. — The application or system software starts the QP cleanup process.
2	FIN Received	FW2, Closing	• Send ACK to the FIN. • Set the TCP state to time wait (note that iWARP state is not changed). • Issue AE_LLP_CLOSE_COMPLETE.	See Row 1 of Software Actions.
3	FIN Received	EST, Terminate	See Row 0 of E810 Actions.	See Row 1 of Software Actions.
4	FIN Received	FW1, Terminate	See Row 1 of E810 Actions.	See Row 1 of Software Actions.
5	FIN Received	FW2, Terminate	See Row 2 of E810 Actions.	See Row 1 of Software Actions.
7	ACK Received	FW1, Closing or Terminate	• Set the TCP state to FW2.	N/A

Table 11-6. iWARP QP State Transition Driven from the Wire [continued]

Row #	Wire Activity	Initial QP State (TCP, iWARP)	E810 Actions	Software Actions
8	ACK Received	FW2, Closing or Terminate	• Send ACK to the FIN. • Set the TCP state to time wait (note that iWARP state is not changed). • Issue AE_LLP_CLOSE_COMPLETE.	See Row 1 of Software Actions.
9	ACK Received	Closing, Closing or Terminate	• Set the TCP state to time wait (note that iWARP state is not changed). • Issue AE_LLP_CLOSE_COMPLETE.	See Row 1 of Software Actions.

11.4.3 RoCEv2 State Management

RoCEv2 QP state is controlled by the application to a large extent. The verbs interface enables the application to manage QP transitions and provides rules for what the RNIC Interface (RI) must enforce.

This section is not intended to repeat all information from the specification. It is intended to be an overview of the expected usage model to be used with the E810. In the end, it is a combination of hardware support from the E810 and supporting software that provides a verbs-compliant interface.

Table 11-7 lists the operations performed by the E810 based on the RoCEv2 QP state.

Table 11-7. RoCEv2 QP State Behavior

QP State	QP Activities
Non-existent	• Doorbell rings are dropped silently. • No transmit (because doorbell rings are dropped). • No receive processing.
Reset	• Software-only state.
Initialized	• RQ WQEs can be posted. • No SQ WQEs can be posted. • No receive processing.
RTR	• RQ WQEs are processed by hardware. • Process and respond to any inbound operations including read requests. • Software prevents SQ WQEs from being posted. • RQ WQEs can be posted.
RTS	• Normal SQ/CQ/CEQ processing.
SQ Drain	• Hardware finishes SQ WQEs that have been started but does not start new WQEs.
SQ Error (RoCEv2 UD only)	• Software-only state.
Error	• No SQ WQEs processed. • No RQ WQEs processed.

QP state transitions and the associated software and hardware/firmware actions driven from software interactions with the application are listed in Table 11-8.

Table 11-8. RoCEv2 QP State Transition

Initial and Next QP State	Software Actions During State Transition	Hardware Actions During State Transition
Nonexistent -> Initialized (CreateQP)	• Initialize the HMC backing pages necessary to support the QP. • Allocate/Initialize all software tracking structures for the PD, CQ, QP. • Assign QP to a PD. • Create CQ. • Allocate SQ and RQ. • Allocate the doorbell shadow area. • Perform parameter checks specified by the verbs specification. • Issue CreateQP WQE to CQP.	• Update QP context.
Initialized -> Initialized	• Perform parameter checks specified by the verbs specification. • Issue Modify QP to CQP, next state = Initialized.	• QP state parameters are updated.
Initialized -> RTR	• Issue Modify QP to CQP, next state = RTR, which allows Receives to be processed. • Software maintains the RTR state so it will prevent Sends from being posted.	• Receives can be processed by hardware.
RTR -> RTS	• Modify the software state to RTS (QP is already in RTS state). • Perform parameter checks specified by the verbs specification.	• Start scheduling work for the QP.
AnyState -> Error	• Perform parameter checks specified by the verbs specification. • Issue Modify QP WQE to CQP with the next state set to error, reset connection (if connection not already reset) and remove quad hash (if quad hash entry is present and has not been previously removed) by setting appropriate bits in the bits set. • If any WQEs are still pending, issue flush WQEs operation to CQP after the Modify QP operation completes. Handle poll for completion requests for flushed WQEs, the CQ has one CQE for each WQ that had pending WQEs, software must report all pending WQEs as flushed once it gets the single flushed CQE (per WQ with pending WQEs), the number of WQEs flushed can be determined from software's head and tail for the SQ and RQ at the point in time that the poll for completion call is made that returns a flushed CQE.	• Set RoCEv2 state to error. • Wait for all packets to clear the pipeline. • Clean up QP context (prevent future scheduling). • When the flush WQEs operation is issued and any WQEs are outstanding on the SQ and/or RQ as requested by host software, complete the first pending WQE for each WQ (flushed completion status).
RTS -> RTS	• Perform parameter checks specified by the verbs specification. • Ensure that the number of currently pending outbound RDMA read requests is less than or equal to the new ORD value. • Ensure the only parameter that is changed is ORD size, it is only allowed to shrink or remain the same. • The ARP index and local MAC Address can also be modified.	• Update the changed value(s).
RTS -> SQD	• Issue Modify QP WQE with the next state set to SQD.	• Hardware completes SQ WQEs that have been started. When these are finished, an AE_QP_SUSPEND_COMPLETE asynchronous event is generated. • Inbound operations continue to be processed.

Table 11-8. RoCEv2 QP State Transition [continued]

Initial and Next QP State	Software Actions During State Transition	Hardware Actions During State Transition
SQD -> RTS	• Issue Modify QP WQE with the next state set to RTS.	• Resume processing on the SQ.
Error -> Initialized	• Software issues destroy and create QP operations to CQP.	• The hardware does not support this transition, but it can be accomplished by software as described in the Software Actions.
Error -> Error	• Submit a Modify QP WQE to CQP with next state = Error. • If any WQEs are still pending, issue flush WQEs operation to CQP after the Modify QP operation completes.	• See RTS -> Error.

QP state transitions and the associated software and hardware/firmware actions driven from wire interactions with remote peer are listed in Table 11-9. Software state transitions are not listed.

Table 11-9. RoCEv2 QP State Transition Driven from the Wire

Row #	Wire Activity	Initial QP State	E810 Actions	Software Actions
0	DREQ message Received	Any state except Error		• Stop processing new post RQ operations if not already stopped. • Issue appropriate events. • If system software does not automatically generate closing, start the closing process based on the received event by issuing Modify QP with the next state set to error.
1	DREQ message Received	Error		• When this message is received: — Stop processing new post RQ operations if not already stopped. — SQ operation processing should have been stopped on the Modify QP with the next state set to closing. — Issue appropriate events. — Issue Flush WQE CQP operation. — If DREP has not been sent, send it now. — The application or system software starts the QP cleanup process.

11.4.4 Exception Queues

Each QP designates a UDA queue to receive exceptions. Some exceptions are iWARP partial FPDU support and CRC error packets received during closing errors. The designated UDA queue receives completions, and software is responsible for taking the appropriate action. For most cases, software logs the error and puts the QP in error state. iWARP Partial FPDUs are handled as described in Section 11.4.4.1.

11.4.4.1 iWARP Partial FPDU Support

The E810 supports partial iWARP FPDUs using software assist. High performance iWARP implementations pay attention to the underlying network MTU and form iWARP FPDUs that do not span Ethernet segments. In the case where software iWARP implementations (or other RNICs) form iWARP FPDUs that span Ethernet segments, the E810 reports these Ethernet packets to the UDA queues. Additionally, the PE writes the *first_partial_sequence_number* field in the Q2 area of the associated QP context. See Section 11.6.1 for the definition of the Q2 area.

Once a partial FPDU has been received, all subsequent packets for that QP are also forwarded to the UDA queue until software has been able to process the packets and send them back to the E810. Host software must process the packets, break them into full FPDUs, and send them back to the E810 through a UDA SQ with the *DOLOOPBACK* bit set. The E810 then processes the packets as if they had come from the wire as full FPDUs. Once all the packets have been sent back to the E810, the QP is automatically set back to normal operation and no further packets are sent to the UDA queues until a new partial FPDU is received.

Each time the Q2 area has an updated expected sequence number; the *Sequence_Update_Toggle* field is toggled. Software must keep track of the last value that it has seen in the Q2 area. If the value does not match, software should update its expected sequence number.

Note: Software must perform the MPA CRC check on all packets that are sent back through to the PE using the *DOLOOPBACK* bit in addition to reforming full iWARP FPDUs. The E810 does not support checking the MPA CRC in hardware on this path through the chip.

The high-level software algorithm used to handle partial FPDUs is shown in Figure 11-21.

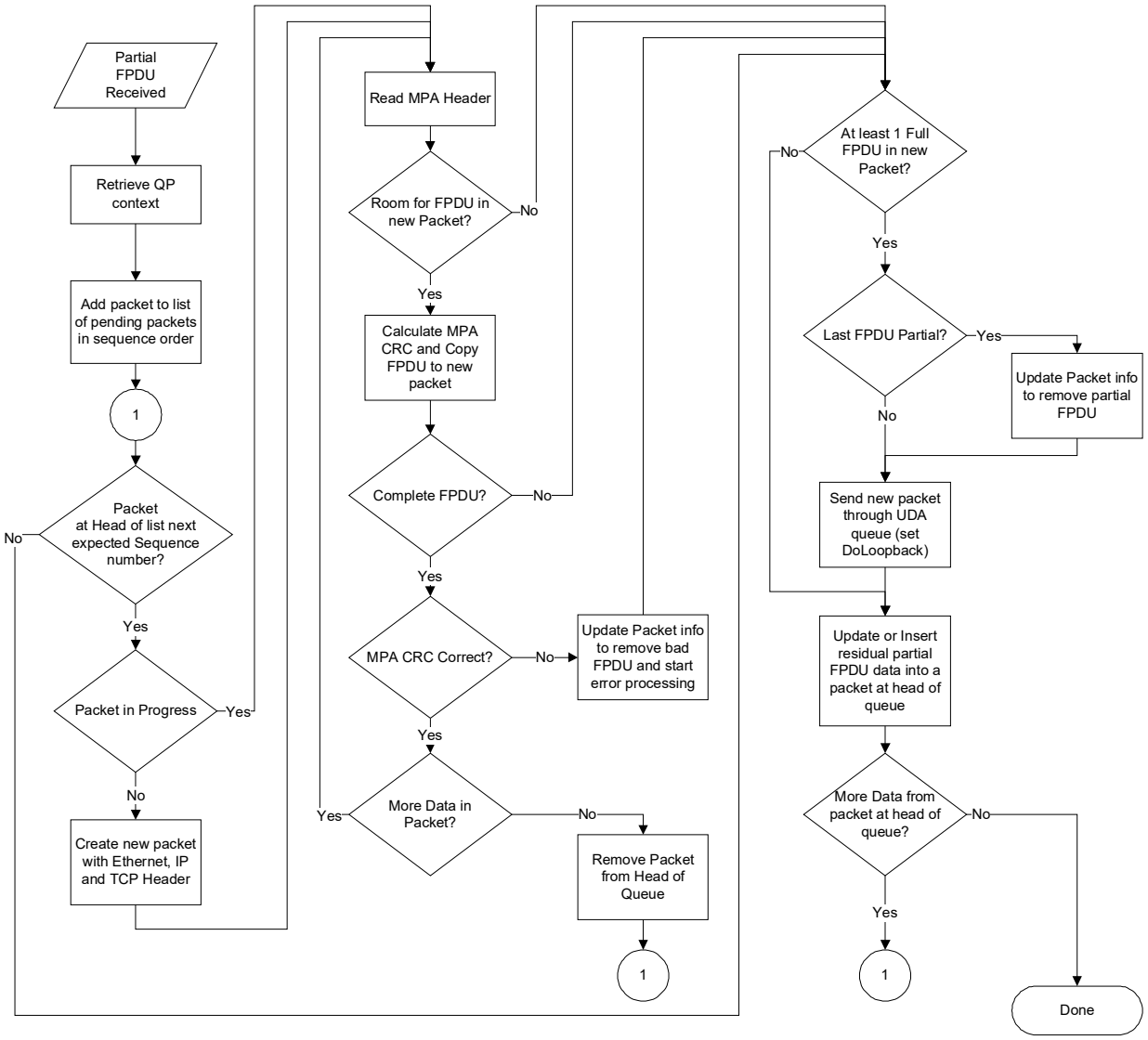


Figure 11-21. Partial FPDU Software Algorithm

11.4.5 Completion Event Queue (CEQ) Entry Format

A CEQ is a circular ring of Completion Event Queue Entries (CEQEs) in host memory. CEQs can be virtually or physically contiguous and are managed using the control QP (see Section 11.5.3.11).

Table 11-10. PE CEQ Entry Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63]	CEQE_Valid	[62:0]	CQ_Context_Value

CEQE_Valid (1 bit)

The *CEQE_Valid* bit for CEQE is a bit that indicates that a CEQE is ready to be processed by software. The polarity of the *Valid* bit changes each time the CEQ wraps from the last entry back to the first entry. This change in polarity reduces software overhead by avoiding the need to clear the *Valid* bit once software has processed a valid CEQE. Software is responsible to clear (set to 0b) all memory in a CEQ initially at CEQ creation. The first iteration (and subsequent odd numbered iterations) through the CEQ, the E810 sets the *Valid* bit to 1b when it writes a new CEQE. For the second iteration (and all even numbered iterations) through the CEQ, the E810 sets the *Valid* bit to 0b when it writes a new CEQE.

CQ_Context_Value (63 bits)

CQ_Context_Value is the value of *CQ_Context_Value* specified at CQ creation (see Section 11.5.3.3). Typically this value is the low 63 bits of a pointer to a software CQ object to enable software to quickly process new CEs.

11.4.6 Asynchronous Event Queue (AEQ) Entry Format

An AEQ is a circular ring of Asynchronous Event Queue Entries (AEQEs) in host memory. AEQs can be virtually or physically contiguous and are managed using the Control QP (see Section 11.5.3.12).

Table 11-11. PE AEQ Entry Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	Completion_Context_Value		
8	[63] [62:61] [60:57] [56:54] [53:50] [49:47]	AEQE_Valid Q2_data_written TCP_State RDMA_State AE_Source RSVD	[46] [45:34] [33] [32:18] [17:0]	QP_CQ_ID_High AE_Code AEQE_Overflow WQ_Desc_Index QP_CQ_ID_Low

Completion_Context_Value (64 bits)

Completion_Context_Value is a software-supplied token that can be used to efficiently locate the software resources necessary to process an AE. The token that is reported can be the *QP_Completion_Context* from the QP that caused the AE, or can be the *CQ_Context_Value* supplied during CQ creation, depending upon the *AE_ID* and *AE_Source* fields of the AE.

WQ_Desc_Index (15 bits)

WQ_Desc_Index value is valid only if the source of the event is a WQ (SQ or RQ) and the *AE_Code* indicates that the *WQ_Desc_Index* field is valid. See Section 11.4.7 for more details on how to determine if *WQ_Desc_Index* is valid.

QP_CQ_ID_Low (18 bits), QP_CQ_ID_High (1 bit)

This field carries either the QP ID that is associated with the AE or the CQ ID that is associated with the AE. The *AE_Source* field (defined later) or the *AE_ID* field are used to determine if the AE is associated with a QP or a CQ.

AE_Code (12 bits)

AE_Code is the AE code the caused the AE. See Section 11.4.7 for more details.

AE_Source (4 bits)

AE_Source defines the source of the AE. The following table lists the values and the related source of the problem.

Value	Description
0000b	Reserved.
00x1b	The AE is associated with the RQ of a QP. *QP_CQ_ID* is the QP ID. *Completion_Context_Value* is set to the value of *QP_Completion_Context* from the associated QP, and *WQ_Desc_Index* is valid.
xx10b	The AE is associated with a CQ. *QP_CQ_ID* is the CQ ID and *Completion_Context_Value* is the *CQ_Context_Value* specified at CQ creation. *WQ_Desc_Index* not valid.
01x1b	*QP_CQ_ID* is the QP ID. *Completion_Context_Value* is set to the value of *QP_Completion_Context* from the associated QP, and *WQ_Desc_Index* is valid. The event is associated with the SQ of a QP.
10x1b	The AE is associated with an inbound RDMA write or inbound RDMA read response operation related to a QP. *QP_CQ_ID* is the QP ID. *Completion_Context_Value* is set to the value of *QP_Completion_Context* from the associated QP, and *WQ_Desc_Index* is not valid.
11x1b	The AE is associated with an outbound RDMA read response operation related to a QP. This value can also be returned is other cases in which the *WQ_Desc_Index* value cannot be retrieved by the E810. *Completion_Context_Value* is set to the value of *QP_Completion_Context* from the associated QP, and *WQ_Desc_Index* is not valid.

Note: Some *AE_ID* values uniquely define the source of the AE. For these events, *AE_Source* should be ignored. See Section 11.4.7 for more details.

RDMA_State (3 bits)

RDMA_State reflects the RDMA state of the connection at the time the AE occurred according to E810 hardware.

Value	Description
000b	Non-existent
001b	Idle
010b	Ready to Send (RTS)
011b	Closing
100b	Reserved
101b	Terminate
110b	Error
111b	Reserved

TCP_State (4 bits)

TCP_State reflects the TCP state of the connection at the time the AE occurred according to E810 hardware. The TCP_State is not valid for RoCEv2 QPs.

Value	Description
0000b	Non-existent
0001b	Closed
0010b	Listen
0011b	Syn_Sent
0100b	Syn_Received
0101b	Established
0110b	Close_Wait
0111b	Fin_Wait_1
1000b	Closing
1001b	Last_Ack
1010b	Fin_Wait_2
1011b	Time_Wait
Note: All other values are reserved.	

AEQE_Valid (1 bit)

The *AEQE_Valid* bit for AEQE indicates that an AEQE is ready to be processed. The polarity of the *Valid* bit changes each time the AEQ wraps from the last entry back to the first entry. This change in polarity reduces software overhead by avoiding the need to clear the *Valid* bit once software has processed a valid AEQE. Software is responsible to clear (set to 0b) all memory in a AEQ initially at AEQ creation. The first iteration (and subsequent odd numbered iterations) through the AEQ, the E810 sets the *Valid* bit to 1b when it writes a new AEQE. For the second iteration (and all even numbered iterations) through the AEQ, the E810 sets the *Valid* bit to a 0b when it writes an new AEQE.

AEQE_Overflow (1 bit)

The *AEQE_Overflow* bit for AEQE indicates that an AEQE has been completely filled and that there might have been a loss of subsequent AEQEs. No further AEQEs are generated until additional space is allocated to the AEQ using the PFPE_AEQALLOC or VFPE_AEQALLOC register.

Q2_data_written (2 bits)

Q2_data_written provides status regarding error information that is reported for errors detected for iWARP connections when an AE is generated. The Q2 data area is configured as part of QP setup. See Section 11.5.3.2 for more information. This values for this field are:

Value	Description
00b	No data associated with this AE has been written to Q2.
01b	Data written to Q2 starts at the Ethernet header of the offending packet.
10b	Data written to Q2 starts at the MPA header of the offending PDU.
11b	Reserved.

11.4.7 AE Codes

Table 11-12 lists the values that can be returned in the *AE_ID* field of an AEQ entry. The column titled "Source" defines if the source of the AE is pre-determined based on the *AE_Code*, or the *AE_Source* field of the AE field must be used to determine the source. Note that the source of the AE effects the *Completion_Context_Value*, *WQ_Desc_Index*, and *QP_CQ_ID* fields in the AE queue entry defined in Section 11.4.6. The *WQ_Desc_Index* field is not valid for AE codes that have a source designated as QP or CQ.

Table 11-12. PE AE Codes (AE_ID)

Asynchronous Event Code	Value	Description	iWARP/ RoCEv2 State after AE	Source	iWARP / RoCEv2 UDA I/R/U
AE_AMP_UNALLOCATED_STAG	0x102	This AE is generated when a memory protection error is detected by the E810. This error indicates that an attempt was made to access a memory region or memory window that is in an unallocated state.	Terminate/ Error	AE_Source	I/R/U
AE_AMP_INVALID_STAG	0x103	This AE is generated when a memory protection error is detected by the E810. This error indicates that an attempt was made to access a memory region or memory window that is in an invalid state.	Terminate/ Error	AE_Source	I/R/U
AE_AMP_BAD_QP	0x104	This AE is generated when a memory protection error is detected by the E810. This error indicates that an attempt was made to access a memory window from a different QP than the QP to which the memory window was bound.	Terminate/ Error	AE_Source	I/R/U
AE_AMP_BAD_PD	0x105	This AE is generated when a memory protection error is detected by the E810. This error indicates that an attempt was made to access a memory region or memory window from a QP with a different PD than the PD associated memory region or window.	Terminate/ Error	AE_Source	I/R/U
AE_AMP_BAD_STAG_KEY	0x106	This AE is generated when a memory protection error is detected by the E810. This error indicates that an attempt was made to access a memory region or memory window when the key portion of the STag does not match the key associated with the memory region or window.	Terminate/ Error	AE_Source	I/R/U
AE_AMP_BAD_STAG_INDEX	0x107	This AE is generated when a memory protection error is detected by the E810. This error indicates that an attempt was made to access a memory region or memory window with an STag index larger than the largest index allowed for the PCI function.	Terminate/ Error	AE_Source	I/R/U
AE_AMP_BOUNDS_VIOLATION	0x108	This AE is generated when a memory protection error is detected by the E810. This error indicates that an attempt was made to access memory outside of the memory area defined by the memory region or memory window.	Terminate/ Error	AE_Source	I/R/U
AE_AMP_RIGHTS_VIOLATION	0x109	This AE is generated when a memory protection error is detected by the E810. This error indicates that an attempt was made to access a memory region or memory window and a rights violation occurred.	Terminate/ Error	AE_Source	I/R/U

Table 11-12. PE AE Codes (AE_ID) [continued]

Asynchronous Event Code	Value	Description	iWARP/ RoCEv2 State after AE	Source	iWARP / RoCEv2 UDA I/R/U
AE_AMP_TO_WRAP	0x10A	This AE is generated when a memory protection error is detected by the E810. This error indicates that an attempt was made to access a memory region or memory window and the tagged-offset plus the ULPDU length caused the address generated for the memory region or window to wrap.	Terminate/ Error	AE_Source	I/R/U
AE_AMP_FASTREG_VALID_ STAG	0x10C	This AE is generated when a memory protection error is detected by the E810. This error indicates that an attempt was made to issue a registration operation to a valid memory region.	Terminate/ Error	AE_Source	I/R
AE_AMP_FASTREG_MW_STAG	0x10D	This AE is generated when a memory protection error is detected by the E810. This error indicates that an attempt was made to issue a registration operation to a memory window.	Terminate/ Error	AE_Source	I/R
AE_AMP_FASTREG_INVALID_ RIGHTS	0x10E	This AE is generated when a memory protection error is detected by the E810. This error indicates that a rights mismatch was detected during a registration operation.	Terminate/ Error	AE_Source	I/R
AE_AMP_FASTREG_INVALID_ LENGTH	0x110	This AE is generated when a memory protection error is detected by the E810. This error indicates that an attempt was made to issue a registration operation with an invalid length.	Terminate/ Error	AE_Source	I/R
AE_AMP_INVALIDATE_SHARED	0x111	This AE is generated when a memory protection error is detected by the E810. This error indicates that an attempt was made to invalidate a shared memory region.	Terminate/ Error	AE_Source	I/R
AE_AMP_INVALIDATE_NO_ REMOTE_ACCESS_RIGHTS	0x112	This AE is generated when a memory protection error is detected by the E810. This error indicates that an attempt was made to remotely invalidate a memory region or memory window that does not have remote access rights specified.	Terminate/ Error	AE_Source	I/R
AE_AMP_INVALIDATE_MR_ WITH_BOUND_WINDOWS	0x113	This AE is generated when a memory protection error is detected by the E810. This error indicates that an attempt was made to invalidate a memory region that still has one or more memory windows bound to it.	Terminate/ Error	AE_Source	I/R
AE_AMP_MWBIND_VALID_ STAG	0x114	This AE is generated when a memory protection error is detected by the E810. This error indicates that the STag used on a bind operation is a memory window that is already valid.	Terminate/ Error	AE_Source	I/R
AE_AMP_MWBIND_OF_MR_ STAG	0x115	This AE is generated when a memory protection error is detected by the E810. This error indicates that the STag used on a bind operation is a memory region instead of a memory window.	Terminate/ Error	AE_Source	I/R
AE_AMP_MWBIND_TO_ZERO_ BASED_STAG	0x116	This AE is generated when a memory protection error is detected by the E810. This error indicates that an attempt was made to bind a memory window to a zero-based memory region.	Terminate/ Error	AE_Source	I/R
AE_AMP_MWBIND_TO_MW_ STAG	0x117	This AE is generated when a memory protection error is detected by the E810. This error indicates that the STag specified for the memory window to be bound to is a memory window instead of a memory region.	Terminate/ Error	AE_Source	I/R

Table 11-12. PE AE Codes (AE_ID) [continued]

Asynchronous Event Code	Value	Description	iWARP/ RoCEv2 State after AE	Source	iWARP / RoCEv2 UDA I/R/U
AE_AMP_MWBIND_INVALID_ RIGHTS	0x118	This AE is generated when a memory protection error is detected by the E810. This error indicates that a rights violation was detected during a memory window bind operation.	Terminate/ Error	AE_Source	I/R
AE_AMP_MWBIND_INVALID_ BOUNDS	0x119	This AE is generated when a memory protection error is detected by the E810. This error indicates that a bounds violation was detected during a memory window bind operation.	Terminate/ Error	AE_Source	I/R
AE_AMP_MWBIND_TO_ INVALID_PARENT	0x11A	This AE is generated when a memory protection error is detected by the E810. This error indicates that the memory region that a memory window attempted to bind to was invalid.	Terminate/ Error	AE_Source	I/R
AE_AMP_MWBIND_BIND_ DISABLED	0x11B	This AE is generated when a memory protection error is detected by the E810. This error indicates that the memory region that a memory window bind operation was attempted on a memory region that has bind support disabled.	Terminate/ Error	AE_Source	I/R
AE_PRIV_OPERATION_DENIED	0x11C	This AE is generated when a privileged operation is attempted on a non-privileged QP. This includes: • Bind operations without bind enabled. • Fast register without fast register enabled. • STag zero without privilege is not enabled.	Terminate/ Error	QP	I/R/U
AE_AMP_INVALIDATE_TYPE1_ MW	0x11D	A Local Invalidate or Inbound Invalidate targeted a Type 1 memory window.	Terminate/ Error	QP	I/R
AE_AMP_MWBIND_ZERO_ BASED_TYPE1_MW	0x11E	A memory window bind operation specified zero-based addressing for the memory window to be bound.	Terminate/ Error	QP	I/R
AE_AMP_FASTREG_INVALID_ PBL_HPS_CFG	0x11F	An unsupported PBL/Host Page Size configuration was requested (1G pages cannot have a 2-level PBLE with 4K pages).	Terminate/ Error	QP	I/R
AE_AMP_MWBIND_WRONG_ TYPE	0x120	A memory window bind specified a non-matching window type.	Terminate/ Error	QP	I/R
AE_AMP_FASTREG_PBLE_ MISMATCH	0x121	Invalid VM Fast Register request to change a physical MR to a virtually mapped MR or vice-versa.	Terminate/ Error	QP	I/R
AE_UDA_XMIT_DGRAM_TOO_ LONG	0x132	This AE is generated by the E810 when the total length of the packet exceeds MSS configured for QP.	Terminate/ Error	QP	U
AE_UDA_XMIT_BAD_PD	0x133	This AE is generated when a memory protection error is detected by the E810. This error indicates that an attempt was made to access an Address Handle from a QP with a different PD than the PD associated Address Handle.	Error	QP	U
AE_UDA_L4LEN_INVALID	0x135	The L4LEN field is not valid, or the payload length is less than L4LEN DWords.	Error	QP	U
AE_BAD_CLOSE	0x201	This AE is generated when in iWARP state = closing and an iWARP PDU is received. If the E810 detects SQ or Q1 work during a Modify QP RTS->Closing then a completion error is returned to the Modify QP request instead of an AE.	Error	QP	I

Table 11-12. PE AE Codes (AE_ID) [continued]

Asynchronous Event Code	Value	Description	iWARP/ RoCEv2 State after AE	Source	iWARP / RoCEv2 UDA I/R/U
AE_RDMAP_ROE_BAD_LLP_ CLOSE	0x202	This AE indicates that there was either SQ or Q1 work pending when a FIN was received.	Terminate	QP	I
AE_CQ_OPERATION_ERROR	0x203	The E810 attempted to generate a CQ entry on a full CQ that has overflow detection enabled.	N/A	AE_Source	I/R/U
AE_RDMA_READ_WHILE_ORD_ ZERO	0x205	This AE is generated when an RDMA read request is issued to a QP that has the Outbound RDMA Read Queue Depth (ORD) set to 0b.	Terminate/ Error	QP	I/R
AE_STAG_ZERO_INVALID	0x206	Detected on an inbound RDMA write, RDMA read response, or RDMA read request (either source or sink). Inbound 0 byte RDMA read and RDMA write operations are not indicated as errors since no host data is accessed.	Terminate/ Error	QP	I/R
AE_IB_RREQ_AND_Q1_FULL	0x207	This AE is generated if an inbound RDMA read is received and the target QP's inbound RDMA RQ (Q1) is full.	Terminate/ Error	QP	I/R
AE_IB_INVALID_REQUEST	0x208	The remote side detected an operation that is outside the established use for the transport service (such as an invalid opcode or length too long). ***Note:*** The error is detected on the remote machine, but the AE is on the local machine.	Error	QP	R
AE_WQE_UNEXPECTED_ OPCODE	0x20A	This AE is generated if QP encounters a WQE with an invalid OP code on an SQ.	Transmit Suspended	QP	I/R/U
AE_WQE_INVALID_PARAMETER	0x20B	This AE is generated if QP encounters a WQE with invalid parameters.	Terminate/ Error	AE_Source	I/R/U
AE_WQE_INVALID_FRAG_DATA	0x20C	This occurs when a WQE has more than 14 fragments or more than 224 bytes of inline data. It also occurs when a WQE crosses a 256-byte boundary.	Terminate/ Error		I/R/U
AE_IB_REMOTE_ACCESS_ ERROR	0x20D	This occurs when the receives a NAK for remote access error.	Error	QP	R
AE_IB_REMOTE_OP_ERROR	0x20E	This occurs when the receives a NAK for remote operational error	Error	QP	R
AE_WQE_LSMM_TOO_LONG	0x220	This AE is generated if the last streaming mode message specified on the Modify QP operation from the iWARP IDLE state to RTS state is longer than the MSS.	Terminate/ Error	AE_Source	I
AE_DDP_INVALID_MSN_GAP_ IN_MSN	0x301	The AE is generated when a gap is detected in the MSN used to index the RQ or Q1 when data is in order with respect to TCP sequence space.	Terminate/ Error	AE_Source	I
AE_DDP_UBE_DDP_MESSAGE_ TOO_LONG_FOR_AVAILABLE_ BUFFER	0x303	This AE is generated on inbound iWARP PDUs if the message offset plus the size of the PDU data is larger than the number of bytes in the RQ WQE targeted by the PDU.	Terminate/ Error	AE_Source	I/R
AE_DDP_UBE_INVALID_DDP_ VERSION	0x304	This AE is generated when an inbound iWARP PDU has an incorrect DDP version number.	Terminate	QP	I
AE_DDP_UBE_INVALID_MO	0x305	This AE is generated if the message offset is larger than the number of bytes in the RQ WQE targeted by the PDU on inbound iWARP PDUs.	Terminate/ Error	QP	I

Table 11-12. PE AE Codes (AE_ID) [continued]

Asynchronous Event Code	Value	Description	iWARP/ RoCEv2 State after AE	Source	iWARP / RoCEv2 UDA I/R/U
AE_DDP_UBE_INVALID_MSN_ NO_BUFFER_AVAILABLE	0x306	This AE is generated if an inbound send operation targets an unallocated RQ WQE and the E810 is configured to drop the connection for this condition.	Terminate	AE_Source	I
AE_DDP_UBE_INVALID_QN	0x307	This AE is generated when an inbound untagged iWARP PDU has an incorrect queue number.	Terminate	QP	I
AE_DDP_NO_L_BIT	0x308	Detected when there is no L bit on an inbound RDMA read request.	Terminate	QP	I
AE_RDMAP_ROE_INVALID_ RDMAP_VERSION	0x311	This AE is generated when an inbound iWARP PDU has an incorrect RDMAP version number.	Terminate	QP	I
AE_RDMAP_ROE_UNEXPECTED _OPCODE	0x312	This AE is generated when an inbound iWARP PDU has an incorrect opcode number.	Terminate/ Error	QP	I/R
AE_ROE_INVALID_RDMA_READ _REQUEST	0x313	This AE is generated when an inbound RDMA read iWARP PDU is received, but the QP is not enabled for inbound RDMA read support.	Terminate/ Error	QP	I/R
AE_ROE_INVALID_RDMA_ WRITE_OR_READ_RESP	0x314	This AE is generated when an inbound RDMA read response or RDMA write iWARP PDU is received, but the QP is not enabled for inbound RDMA read responses or RDMA writes.	Terminate/ Error	QP	I/R
AE_RoCE_RSP_LENGTH_ ERROR	0x316	The RoCEv2 responder received a packet with an RETH and the *dma_length* field is inconsistent with the payload length.	Error	QP	R
AE_INVALID_ARP_ENTRY	0x401	This AE is reported when a connection attempts to transmit a packet using an invalid ARP entry. Host software either enables the QP for transmitting after updating the ARP entry, or uploads/destroys the connection because the ARP entry is no longer valid. In the case of updating the ARP entries, statistics report can be off by the number of packets/octets/ messages that were attempted to be transmitted while the ARP entry was invalid.	Transmit Suspended	QP	I/R/U
AE_INVALID_TCP_OPTION_ RCVD	0x402	This AE is generated when the E810 encounters an unsupported TCP option and the E810 is not configured to ignore unsupported TCP options via the *ignore_tcp_uns_options* QP context bit. The packets with invalid TCP options are dropped if this AE is generated.	No Change	QP	I
AE_INVALID_AH_ENTRY	0x406	This AE indicates that the AH was not valid.	Error	QP	R
AE_LLP_CLOSE_COMPLETE	0x501	This AE indicates that the TCP state of an iWARP QP has transitioned to either closed or time wait in a graceful fashion.	No Change	QP	I
AE_LLP_CONNECTION_RESET	0x502	This AE indicates that a TCP packet with the *RST* bit set has been received on a TCP connection that is associated.	Error	QP	I
AE_LLP_FIN_RECEIVED	0x503	This AE indicates that a TCP packet with the *FIN* bit set has been received and the iWARP state is RTS (implies TCP state is established). Host software is responsible for issuing a Modify QP -> Closing to complete the QP transition to the closing the state.	No Change	QP	I

Table 11-12. PE AE Codes (AE_ID) [continued]

Asynchronous Event Code	Value	Description	iWARP/ RoCEv2 State after AE	Source	iWARP / RoCEv2 UDA I/R/U
AE_LLP_RECEIVED_MPA_CRC_ ERROR	0x505	This AE is reported when an MPA or ICRC Error CRC error is detected by software using the AE code capability of the CQP Flush WQEs operation.	Terminate/ Error	QP	I/R
AE_LLP_SEGMENT_TOO_SMALL	0x507	This AE is reported when an iWARP segment is received that is too small to contain a DDP or RDMAP header.	Terminate/ Error	QP	I
AE_LLP_SYN_RECEIVED	0x508	This AE indicates that a TCP packet with the *SYN* bit set is received on a connection that is associated with an iWARP connection.	Error	QP	I
AE_LLP_TERMINATE_RECEIVED	0x509	This AE indicates that a terminate message has been received for an iWARP QP. The received terminate data is placed in the Q2 area associated with the QP in host memory.	Terminate	QP	I
AE_LLP_TOO_MANY_RETRIES	0x50A	This AE is used to report too many retransmission retries.	Transmit Suspended	QP	I/R
AE_LLP_DOUBT_REACHABILITY	0x50C	This AE is used to report that it has reached the doubt reachablity threshold for the given connection. This is used in dead gateway detection or black holes.	No Change	QP	I/R
AE_LLP_CONNECTION_ ESTABLISHED	0x50E	This AE is used to report that a connection has been established. It occurs when the first iWARP message has been received.	RTS	QP	I
AE_RESOURCE_EXHAUSTION	0x520	This AE is used to report that a QP attempted to allocate Q1 or XMIT FIFO resource and no resource was available.	No Change	AE_Source	I/R
AE_RESET_SENT	0x601	This AE is generated when the E810 sends a reset after host software has requested the connection to be reset via a Modify QP operation.	No Change	QP	I
AE_TERMINATE_SENT	0x602	This AE is generated when the E810 sends a terminate message after host software has requested a terminate to be sent on a connection via a Modify QP operation.	No Change	QP	I
AE_RESET_NOT_SENT	0x603	This AE is generated when the E810 did not send a reset after host software has requested the connection to be reset via a Modify QP operation. The reset is not sent due to TCP state already having been transitioned to time wait or closed.	No Change	QP	I
AE_LCE_QP_CATASTROPHIC	0x700	This AE is generated by the E810 when it receives an error on a hardware transaction.	Terminate/ Error	QP	I/R/U
AE_LCE_FUNCTION_ CATASTROPHIC	0x701	This AE is generated by the E810 when it receives an error on a hardware transaction that impacts the entire PCI function. QP_CQ_ID is not reliable for this AE and should be ignored.	Terminate/ Error	N/A	I/R/U
AE_LCE_CQ_CATASTROPHIC	0x702	This AE is generated by the E810 when it receives a local catastrophic error on a hardware transaction that impacts a CQ.	Terminate/ Error	CQ	I/R/U
AE_QP_SUSPEND_COMPLETE	0x900	This AE is generated upon completion of the requested QP Suspend operation.	Transmit Suspended	QP	I/R

11.4.8 Steering Tag (STag) and Processing Hint Support for PE Traffic (TPH)

See Section 3.1.2.6.2 for information on how to enable TLP processing hint support.

Table 11-13 lists how the STag and processing hints are generated and how TPH operation is enabled for different types of DMA traffic associated with the PE.

Table 11-13. STag and Processing Hint Programming

Traffic Access	STag Location	PH Value	Enable
CQ Element Writes	CQ Context *TPH_value*	GL_TPH_CTRL.*Data_PH*[1]	CQ Context *TPH_en*
CQ Doorbell Shadow Area Reads	CQ Context *TPH_value*	GL_TPH_CTRL.*Data_PH*[1]	CQ Context *TPH_en*
CEQ Element Writes	CEQ Context *TPH_value*	GL_TPH_CTRL.*Data_PH*[1]	CEQ Context *TPH_en*
SQ WQ Element Reads	QP Context *SQ_TPH_value*	GL_TPH_CTRL.*Data_PH*[1]	QP Context *SQ_TPH_en*
SQ Work Doorbell Shadow Area Writes	QP Context *SQ_TPH_value*	GL_TPH_CTRL.*Data_PH*[1]	QP Context *SQ_TPH_en*
RQ WQ Element Reads	QP Context *RQ_TPH_value*	GL_TPH_CTRL.*Data_PH*[1]	QP Context *RQ_TPH_en*
Data Payload Writes	QP Context *RQ_TPH_value*	GL_TPH_CTRL.*Data_PH*	QP Context *RCV_TPH_en*
Data Payload Reads	QP Context *SQ_TPH_value*	GL_TPH_CTRL.*Data_PH*[1]	QP Context *XMIT_TPH_en*

1. Default is 10b (target).

The E810 has TPH values for CQ and CEQ in each of the various contexts. When a read is done (such as reading the CQ shadow area), the E810 updates the context with the current TPH hint that is returned in the completion (CQ:*TPH_value*). If the enable bit is set in the particular context, the E810 should provide the TPH information on all subsequent PCI transactions associated with CQ or CEQ transactions. There is no TPH value used on AEQE writes because they are so infrequent that it is unlikely to ever get them on the correct CPU.

QP context has two values for TPH: *SQ_TPH* and *RQ_TPH*. They are independent of each other because the SQ and RQ could be on different processors in PE accelerations. They also have separate enable bits.

11.5 Resource Management

PE resources are managed through a combination of the HMC Function Private Memory (FPM) space (described in Section 9.3) and Control QP operations (described in the following sections).

Figure 11-22 shows the PE-related data structures that are located in host memory and that reside on-die for the E810. Before any QP, or CQ operation is attempted with CQP, the associated HMC pages must be initialized and allocated as specified in Section 9.3.7. Once all FPM Page Descriptor pages and backing pages have been properly allocated, initialized, and allocated to the E810, CQP can be used to create/modify/destroy various objects necessary for PE operations. Note that Figure 11-22 only depicts memory that is visible to the E810. Many other software allocated data structures are allocated to track progress of QPs, CQs, an so on, as shown in Section 11.4.1. All of the host memory shown in Figure 11-22 must be pinned memory. QP and CQ memory including the doorbell shadow areas might be allocated from kernel or userspace virtual memory. However, all other memory shown must be allocated from the kernel to provide the proper level of protection between different processes accessing the E810 concurrently.

Note: Any reset that affects a single PCI function also resets the HMC and require reprogramming of all HMC resources. Global resets also reset the segment descriptor programming that require re-selecting the HMC resource profile if the default profile was not in use.

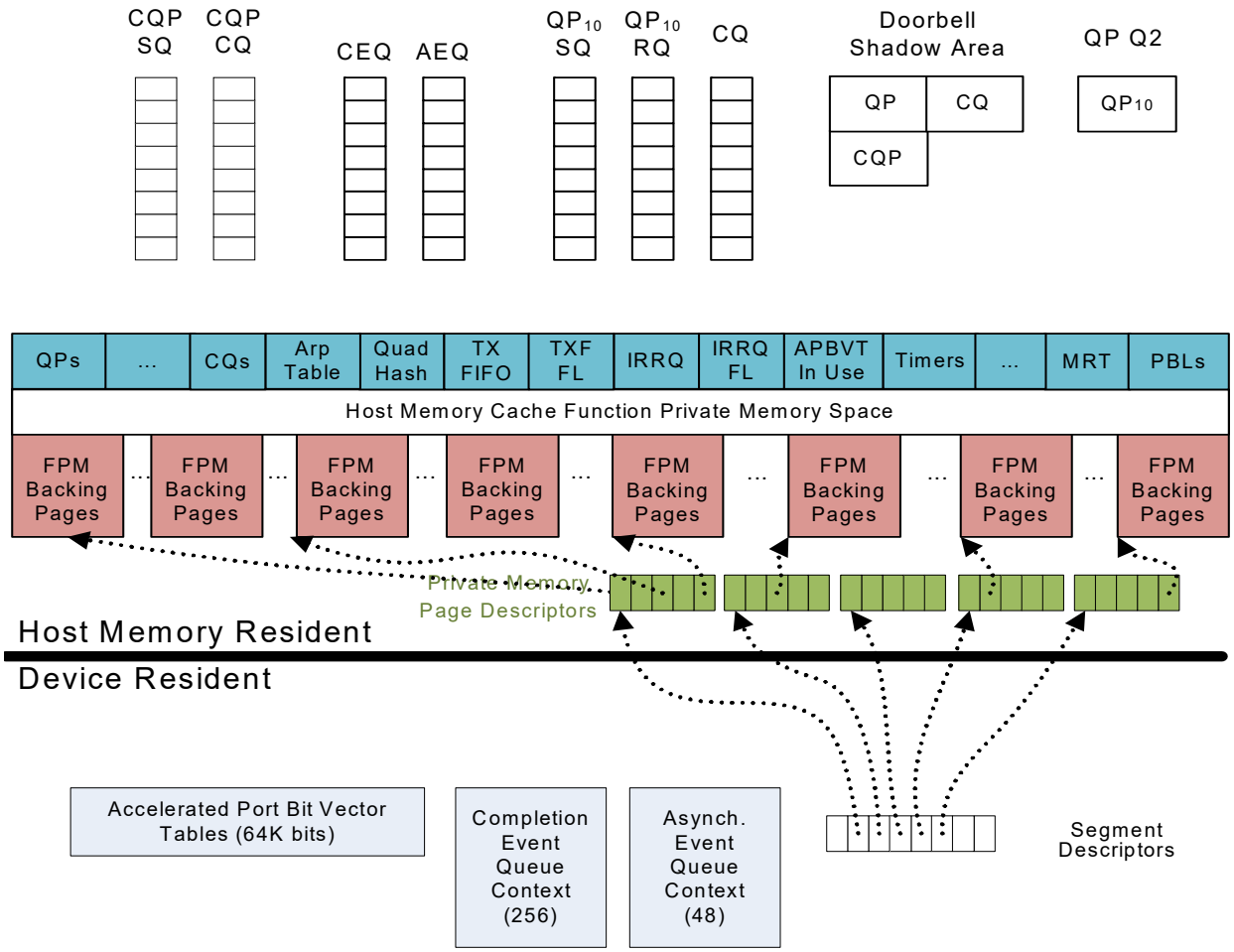


Figure 11-22. E810 and Host Memory Data Structures

FPM page descriptors and backing pages can be allocated on the fly in order to reduce the kernel drivers memory footprint. However, certain objects must be allocated at driver initialization.

Table 11-14. PE Objects Dependencies

PE HMC Object	HMC Object Page Population Algorithm	Object Dependencies	Non-HMC Object Dependences
QP Context	Dynamic	• ARP Table Entries • Quad Hash Entries • Tx FIFO Entries • APBVT In-Use Table • CQ • Physical Buffer List Entries if Virtually Mapped	• VSI • Local MAC Address • L2 Tag Configuration
CQ Context	Dynamic	• Physical Buffer List Entries if Virtually Mapped	
ARP Table Entries	Dynamic	None	
Quad Hash Entries	Static	None	
Tx FIFO Entry Free List	Static	None	
Tx FIFO Entries	Static	• Tx FIFO Entry Free List	
Inbound Read Request Queue Entry Free List	Static	None	
Inbound Read Request Queue Entries	Static	• IRRQ Entry Free List	
Accelerated Port Bit Vector Table In-Use Table	Static	None	
Multicast Groups	Dynamic	None	
Address Handles	Dynamic	None	
Memory Registration Table	Dynamic	None	
Physical Buffer List Entries	Dynamic	None	
Read Response FIFO Entries Free List	Static	None	
Read Response FIFO Entries	Static	• Read Response FIFO Entries Free List	
Header	Static	• Metadata	
Metadata	Static	• Header	
Out of Order Send Completion (OOISC) FIFO	Static	• Out of Order Send Completion (OOISC) FIFO Free List	
Out of Order Send Completion (OOISC) FIFO Free List	Static	None	

All PE objects marked as "Static" in Table 11-14 must be allocated at driver initialization time. Entries marked as "Dynamic" can be allocated and deallocated on an as-needed basis. Deallocation of HMC FPM page descriptor pages and backing pages is described in Section 9.3.8. Objects shown as dependencies must be populated and allocated via CQP (along with the HMC FPM memory for the object itself) prior to issuing a CQP operation to manipulate the PE object. More information regarding the object dependencies is available in the individual CQP operations in following sections.

Note: PF resets clears the PE state including any HMC configuration settings for the PF and all associated VFs. VF resets clear all PE state and HMC configuration settings only for the VF if the VF is enabled for PE accelerations. See Table 4-1 on page 298 for more information on reset sources and the affected internal logic.

Dynamic objects of the same type can share a cache line. However, a dynamic object must not share any cache lines with another object that has a different type.

11.5.1 PE Initialization

Table 11-15 lists the high-level steps required to initialize a PE. Table 11-16 and Table 11-17 list the high-level steps required to bring an RDMA or UDA QP to the operational state for the E810. These lists are not meant to be exhaustive, but more as a guide to show examples of a typical initialization flow. It is assumed that the LAN initialization of the VSI has enabled PE operation and has the LAN queues operational for the PF.

Table 11-15. PE Resource Initialization

Step	Resource	Responsible Software Component	Action(s)	Notes
1	Determine the IP and MAC Addresses for the associated LAN queue pool.	E810 driver.	Each PE interface is associated with a specific LAN interface. The mechanism to find the interface information is operating system specific.	
2	Enable RDMA/UDA for the corresponding VSI.	E810 driver.	Enable the RDMA Packet Profiles for RDMA in this VSI.	
3	Enable Local VSI Loopback.	E810 driver.	PE enables communication between QPs associated with the same VSI. Set *ALLOWLOOPBACK* and *ALLOWLOCALLOOPBACK* bits in VSI_SRCSWIDCTRL register.	This step is not required. The PE uses SWPE to perform loopback.
4	Setup the PE portion of the HMC.	E810 driver.	Wait for PE firmware to be initialized by polling GLPE_CPUSTATUS0 is set to 0x80. First the HMC base and size registers must be programmed for this PF. Subsequently appropriate backing pages must be initialized.	See Section 9.3 for details on initializing the HMC.
5	Acquire one or more MSI-X vectors for the PE.	E810 driver.	MSI-X vectors can be shared between LAN and protocol operation or can be partitioned to enable independent operation.	See the previous section on LAN initialization for more information on MSI-X vector initialization.
6	Initialize the PE control QP.	E810 driver.	Allocate memory for the CQP SQ and CQP context and write to the PECCQPHIGH and PECCQPLOW registers.	See Section 11.5.2.1 for more details.
7	Initialize the PE CQ 0.	E810 driver.	Allocate memory for the CQ and issue a create CQ operation to CQP.	See Section 11.5.3.3 for details on the Create CQ CQP operation.
8	Initialize the PE CEQ 0.	E810 driver.	Allocate memory for the CEQ and issue a create CEQ operation to CQP, The CEQ is associated with a E810 interrupt.	See Section 11.5.3.11 for details on the create CEQ CQP operation.
9	Initialize the PE AEQ.	E810 driver.	Allocate memory for the AEQ and issue a create CEQ operation to CQP.	See Section 11.5.3.12 for details on the create AEQ CQP operation.

Table 11-16 lists steps required to bring RDMA QP to the operational state. PE initialization steps are common for RDMA and UDA traffic and should be done only once.

Table 11-16. RDMA Resource Initialization

Step	Resource	Responsible Software Component	Action(s)	Notes
1	Create a Protection Domain.	E810 driver on behalf of a verbs application.	Reserve a doorbell page from the PE doorbell pages in the E810 BAR.	See Section 11.4.1.5.1 for more information on doorbell pages.
2	Allocate a CQ.	E810 driver on behalf of a verbs application.	Allocate memory for the CQ and issue a create CQ operation to CQP. The CQ is associated with the CEQ from Step 7.	See Section 11.5.3.3 for details on the create CQ CQP operation.
3	Allocate a QP.	E810 driver on behalf of a verbs application.	Allocate memory for the QPs SQ and RQ and issue a Create QP operation to CQP.	See Section 11.5.3.2 for details on the create QP CQP operation.
4	Allocate Memory Regions.	E810 driver on behalf of a verbs application.	Allocate physical buffer list entry ranges for the page list that backs the memory provided by the application and then issue register memory region CQP operation.	See Section 11.5.3.4 for more information on the register memory region operation and also Section 11.4.1.4 for an overview of the E810's memory registration capabilities.
5	Post Work Requests to the QPs RQ.	E810 driver on behalf of a verbs application.	Fill in a RQ work request with the information supplied by the application and submit the work.	See Section 11.6.6.2 for information about the RQ work request format and Section 11.4.1.5.8 for more information on RQ operation.
6	Post Work Requests to the QPs SQ.	E810 driver on behalf of a verbs application.	Fill in a SQ work request with the information supplied by the application and submit the work.	See Section 11.6.6.1 for information about the SQ work request format and Section 11.4.1.5.2 for more information on SQ operation.
7	Process Interrupts and the CEQ (optional).	E810 driver.	Once an interrupt occurs for a CEQ, the CQ can be determined from the CEQ Entry. Events are generated to that application to let the application know that an event for a specific CQ has been received.	See Section 11.4.1.2 for more information on processing CEQ entries.
8	Process the CQ (optional).	E810 driver on behalf of a verbs application.	Once the application has either received a completion event or is constantly polling the CQ, CQ entries are processed by the application.	See Section 11.4.1.3 for more information on processing CQ entries.

Table 11-17 lists steps required to bring UDA QP to the operational state. PE initialization steps are common for RDMA and UDA traffic and should be done only once.

Table 11-17. UDA Resource Initialization

Step	Resource	Responsible Software Component	Action(s)	Notes
1	Create a Protection Domain.	E810 driver on behalf of a verbs application.	Reserve a doorbell page from the PE doorbell pages in the E810 BAR.	See Section 11.4.1.5.1 for more information on doorbell pages.
2	Allocate a CQ.	E810 driver on behalf of a verbs application.	Allocate memory for the CQ and issue a create CQ operation to CQP. The CQ is associated with the CEQ from Step 7.	See Section 11.5.3.3 for details on the create CQ CQP operation.
3	Allocate a QP.	E810 driver on behalf of a verbs application.	Allocate memory for the QPs SQ and RQ and issue a create QP operation to CQP.	See Section 11.5.3.2 for details on the create QP CQP operation.
4	Allocate Memory Regions.	E810 driver on behalf of a verbs application.	Allocate PBL entry ranges for the page list that backs the memory provided by the application and then issue register memory region CQP operation.	See Section 11.5.3.4 for more information on the register memory region operation and also Section 11.4.1.4 for an overview of the E810's memory registration capabilities.
5	Allocate Address Handle (optional).	E810 driver on behalf of a verbs application.	For each destination Node issue a create address handle operation to CQP. Allocated address handle is used to generate Ethernet/IP headers.	See Section 11.5.3.16 for more information on the create address handle operation and also Section 11.5.3.6 for an overview of the E810's address handle capabilities.
6	Bind to Local Port (optional).	E810 driver on behalf of a verbs application.	For each local port allocated for UDP traffic issue a management quad hash entry operation to CQP. This operation will enable forwarding of unicast packet to the QP.	See Section 11.5.3.20 for more information on the manage quad hash operation.
7	Post Work Requests to the QPs RQ.	E810 driver on behalf of a verbs application.	Fill in a RQ work request with the information supplied by the application and submit the work.	See Section 11.6.6.2 for information about the RQ work request format and Section 11.4.1.5.8 for more information on RQ operation.
8	Post Work Requests to the QPs RQ.	E810 driver on behalf of a verbs application.	Fill in a SQ work request with the information supplied by the application and submit the work.	See Section 11.6.6.1 for information about the RQ work request format and Section 11.4.1.5.2 for more information on RQ operation.
9	Process Interrupts and the CEQ (optional).	E810 driver.	Once an interrupt occurs for a CEQ, the CQ can be determined from the CEQ Entry. Events are generated to that application to let the application know that an event for a specific CQ has been received.	See Section 11.4.1.2 for more information on processing CEQ entries.
10	Process the CQ (optional).	E810 driver on behalf of a verbs application.	Once the application has either received a completion event or is constantly polling the CQ, CQ entries are processed by the application.	See Section 11.4.1.3 for more information on processing CQ entries.

11.5.2 Control QP (CQP) Operation

CQP operations are modeled after verbs SQ operations as far as the work submission process. Some differences exist for CQP when compared to a typical PE QP:

- CQP QP context is located on-die instead of in the HMC.
- The doorbell register used for work submission is PFPE_CQPDB instead of PFPE_WQEALLOC.
- PFPE_CQPDB take SQ head as input and there is no QP ID specified since each PE enabled PCI function has a dedicated CQP.
- There is no doorbell shadow area associated with CQP.
- Hardware tail can be read from the PFPE_CQPTAIL, but in general software can just track SQ tail based on completion processing from CQ0.

The operations defined for CQP are listed in Table 11-21. CQ0 is the CQ associated with the CQP for each function. All other CQs are available for accelerated PE use.

11.5.2.1 Creating the CQP

CQP is created by first allocating memory for CQP context shown in Section 11.5.2.3 and initializing context. The HMC resource profile is also able to be selected during the CQP creation process.

Note: HMC resource profile is selected during first CQP creation on a device and is then locked. Subsequent attempts to change the HMC resource profile are ignored until all CQPs created on the device have been destroyed.

The resource profile that has been selected is reported in the PFPE_CCQPSTATUS register after CQP has been successfully created. Additionally, the SQ memory and FPM configuration buffer must also be allocated. This memory must be pinned and the physical address must be determined.

Once software has filled in the CQP context properly, the physical address of CQP context is written to the PFPE_CCQPHIGH and PFPE_CCQPLOW registers to notify the E810 of the create CQP operation. Both registers must be written, and the PFPE_CCQPLOW register must be written last. The E810 then fetches the context and create CQP for the current PCI function.

Once CQP has be successfully created, the *CCQP_DONE* bit in the PFPE_CCQPSTATUS register is set to 1b. If an error was encountered while attempting to create CQP, the CCQP_ERR bit is set instead of the CCQP_DONE bit. The major and minor error codes are available in the PFPE_CQPERRCODES register. The next six operations necessary to get CQP to a functional state are to:

1. Query FPM Values.
2. Commit FPM Values.
3. Initialize the HMC resources for CQ0.
4. Create CQ 0.
5. Create CEQ.
6. Create the AEQ.

The operations must be done in this order or unexpected results might be observed. In addition to the CQ0 HMC backing pages initialized in Step 3, if either the CQ or CEQ need to be virtually mapped, the PBLE HMC object backing pages must also be populated before creating the CQ and/or CEQ.
RDMA-enabled VFs use the VFPE_* versions of the registers previously mentioned. These steps involve issuing standard CQP operations with the following exceptions:

- PFPE_CQPTAIL must be polled until the create CQ operation has been submitted to CQP, and only a single CQP operation must be submitted at a time to CQP while polling CQP Tail. Any errors encountered via CQP is reported via the CQP_ERR bit in the PFPE_CQPTAIL register. The major and minor error codes are available in the PFPE_CQPERRCODES register.
- CQ0 can be polled for the completion of the create CQ operation. No events can be generated since there is no CEQ created at the point when CQ0 is created.

One additional CQP operation must be issued anytime after Step 3 but before any iWARP QPs can be created. The static HMC pages allocated operation must be submitted after all static HMC objects have been allocated and configured for the PCI function. See Table 11-14 for the list of statically configured HMC objects.

11.5.2.2 Destroying the CQP

Once all PE resources have been destroyed and deallocated, the control QP related resources much also be destroyed. The flow is the following:

1. Issue a CQP operation to destroy the AEQ.
2. Issue a CQP operation to destroy the CEQ(s).
3. Poll CQ0 to verify these operations completed.
4. Issue a CQP operation to destroy CQ0 (CQP's CQ).
5. Poll CQP tail for completion of the destroy CQ.
6. Destroy CQP by writing the PFPE_CCQPHIGH and PFPE_CCQPLOW registers with 0b.
7. Poll the PFPE_CCQPSTATUS register until it changes to 0b.
8. Destroy the CQ0 HMC resources to ensure that no further accesses to host memory is performed.

Once CQP is destroyed, CQ0 is destroyed as well.

11.5.2.3 CQP Context

CQP context describes the SQ for the control QP.

Table 11-18. CQP Context Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:40] [39:32] [31:24] [23:12] [11:8]	RSVD num_CEQs_per_VF StructVersion RSVD SQ_Size	[7:5] [4:3] [2] [1] [0]	RSVD protocol_used RoCEv2_RTO_Policy Disable_FPDU_Packing Enable_Fine_Grained_Timers
8	[63:9]	SQ_Base	[8:0]	RSVD
16	[63:38] [37:32] [31:8]	RSVD PEEnabledVfCount RSVD	[7:4] [3] [2:0]	rdpa_assist remote_endpoint_trk_en HMCProfileType
24	[63:0]	QP_Completion_Context		
32	[63:32] [31:16]	RSVD hw_major_version	[15:0]	hw_minor_version
40	[63:0]	RSVD		
48	[63:0]	RSVD		
56	[63:0]	RSVD		

StructVersion (8 bits)

Used to specify changes in CQP context format. The only value defined is 0.

Enable_Fine_Grained_Timers (1 bit)

This bit enables fine-grained TCP timer mode for the PCI function associated with the CQP to be created. Fine-grained TCP timer mode changes the PE TCP timers from normal values (2 minutes to ~100 ms) to a more aggressive set of values (~1 s to ~100 μs). Fine-grained TCP timer mode is useful for environments with low packet congestion and/or low packet loss.

SQ_Size (4 bits)

This field encodes the maximum number of work requests for each WQ. The encoding of the SQ sizes are $4*2^{SQ\_Size}$ in terms of 32-byte quanta of memory. Since each CQP WQE takes 64 bytes (as if it consumed an additional fragment descriptor), only the following settings are valid for CQP (all other settings are reserved and might cause unexpected results):

Value	Description
0001b	4 WQEs
0010b	8 WQEs
0011b	16 WQEs
0100b	32 WQEs
0101b	64 WQEs
0110b	128 WQEs
0111b	256 WQEs
1000b	512 WQEs
1001b	1024 WQEs

Value	Description
1010b	2048 WQEs
Note: All other values are reserved.	

Software can only submit N-1 WQEs to a WQ without processing completions for the WQ without exposing the possibility of a WQ overflow. WQ overflow results in indeterministic behavior for the affected WQ.

SQ_Base (55 bits)

SQ_base is the most significant bits of the physical address of the SQ for the control QP. The WQ base address must be evenly divisible by 512 for proper behavior.

HMCProfileType (3 bits)

This field specifies the HMC resource profile to be activated.

Value	Description
001b	Default
010b	SR-IOV VF Primary
011b	SR-IOV Even Distribution
Note: All other values are reserved.	

Specifying a reserved HMC resource profile causes the current HMC resource profile to remain active. This field is ignored for VFs.

remote_endpoint_trk_en (1 bit)

This field enables/disables endpoint tracking. This is experimental.

Value	Description
0b	Disabled
1b	Enabled

PEEnabledVfCount (6 bits)

Only valid when *HMCProfileType* is one of the SR-IOV profile types. Specifies the number of PE enabled VFs to be allocated. Values larger than 32 are rounded down to 32. A value of zero forces the default profile to be selected. This field is ignored for VFs.

See Section 9.3.3, "Private Memory Space Profiles" for more information on HMC resource profiles.

QP_Completion_Context (64 bits)

QP_Completion_Context is reported in every CQ Entry related to the CQP.

Disable_FPDU_Packing (1 bit)

Disables transmission of packets with more than one FPDU per packet for iWARP Queue Pairs.

RoCEv2_RTO_Policy (1 bit)

Value	Description
0b	The RoCEv2 retransmission timer stays constant.
1b	The RoCEv2 retransmission timer doubles on every retransmission up to a maximum value.

protocol_used (2 bits)

Selects which protocol is used.

Value	Description
00b	Any
01b	iWARP
10b	RoCEv2
11b	Reserved

num_CEQs_per_VF (8 bits)

This specifies the max CEQs needed for each VF. The purpose of this field is to limit the number of CEQs a VF uses in the even distribution profile to leave enough CEQs for the PF. A value of 0 preserves the behavior of evenly allocating the CEQs across all functions. This field is valid only if SR-IOV is enabled and the balanced profile is in use.

hw_major_version, hw_minor_version (16 bits each)

Version of hardware that the software driver supports.

Major version values are:

Value	Description
0	E810
Note: All other values are reserved.	

rdpa_assist (4 bits)

This specifies the rate of assist: 0x0=none, 0xF=max.

11.5.2.4 CQP Error Codes

CQP can return a number of error and status code the operations that it performs. Table 11-19 lists these error and status codes that can be returned in the CQ entry listed in Table 11-20.

Table 11-19. CQP Error Codes

Major Error Code	Minor Error Code	Completion Reason	Description
0x0000	0xkk00	STag Valid	The STag queried is in the valid state, where kk is the STag key.
0x0000	0x0001	STag Invalid	The STag queried is in the invalid state.
0x0000	0x0002	RQ WQE Flushed	A RQ WQE was flushed as a result of the flush WQEs operation.
0x0000	0x0003	SQ WQE Flushed	A SQ WQE was flushed as a result of the flush WQEs operation.
0x0000	0x0004	SQ and RQ WQEs Flushed	Both a RQ WQE and a SQ WQE were flushed as a result of the flush WQEs operation.
0x0000	0x0005	Suspend Pending	A suspend QP WQE was issued referencing a busy QP. An AE_QP_SUSPEND_COMPLETE AE is generated when the QP has been successfully suspended.
0xF000	0x000x	Object Cache Error	An object cache error has reported to CQP during this operation. Object cache errors reported in x include: 0: Cache Address Translation Error
0xF001	0x000x	Context Cache Error	A context cache error was reported to CQP during this operation. Context cache errors include: 0x1 – PMAT Error during normal operation 0x2 – PMAT Error during reset 0x4 - Context CRC Error
0xFFFF	0x3000	Packet Count Error	A packet count issue has been detected on a QP that is either being destroyed or uploaded. The state of the QP is indeterminate once this completion code has been observed.
0xFFFF	0x3001	Scratchpad Flush Error	CQP has timed out waiting for TEP to flush scratchpads for a QP that is getting destroyed or uploaded. This error has never been observed and is not expected.
0xFFFF	0x3002	Bad Queue Set Handle	The given queue set handle is associated with a work scheduler node that was not properly enabled.
0xFFFF	0x3003	Invalid WS Node Weight	Returned by Manage Work Scheduler Node if an invalid weight is provided for the specified node_type and/or priority_type.
0xFFFF	0x3004	Invalid Traffic Class	Returned by Manage Work Scheduler Node if the given traffic class is invalid for the current system configuration
0xFFFF	0x4000	Memory Window Bound	The destroy QP operation failed because there are still MWs bound to the QP.
0xFFFF	0x6000	Insufficient Resources (General)	There is an insufficient amount of the resources needed to complete this operation.
0xFFFF	0x6001	Insufficient FLM Resources	A create QP was attempted but there were no available transmit of Q1 FIFO entries.
0xFFFF	0x6002	Insufficient Doorbell Resources	Returned by commit FPM (if requested number of QPs or CQs is greater than count allotted in partition registers).
0xFFFF	0x6003	Hash Filter Programming Failed	An ADD or REMOVE of a hash filter entry was attempted but was not successful.
0xFFFF	0x6004	PCIe Unsupported Request	Access of a software-provided host physical address resulted in an Unsupported Request response from PCIe.
0xFFFF	0x8000	ID Too Big	The ID specified on the operation is too large for the on-board memory configuration.

Table 11-19. CQP Error Codes [continued]

Major Error Code	Minor Error Code	Completion Reason	Description
0xFFFF	0x8002	Invalid State	The operation requested is not valid for this resource based upon its current state.
0xFFFF	0x8003	Invalid Next QP State	The next state specified on the Modify QP operation is not valid based on the QP's current state.
0xFFFF	0x8004	Invalid Queue Size	The SQ, RQ, or Q1 is too large or to small.
0xFFFF	0x8005	Invalid QP Type	The operation is not valid for this type of QP.
0xFFFF	0x8006	No WQE Pending	A flush operation was attempted but there was not a WQE pending.
0xFFFF	0x8007	Bad Close	A Modify QP from RTS->Closing was attempted while there was SQ and/or Q1 activity in progress.
0xFFFF	0x8009	LLP Closed	Returned by Modify QP (next RDMA state = Terminate if the TCP state is already closed or time wait).
0xFFFF	0x800A	Reset Not Sent	Returned by Modify QP (if the TCP state is already closed).
0xFFFF	0x800C	SD Index Out of Range	An attempt has been made to update a VF's page descriptor using an SD that does not reference a PBLE object.
0xFFFF	0x800D	PD Index Out of Range	An attempt has been made to update a VF's page descriptor using a SD and PD index that does not reference a PBLE object.
0xFFFF	0x800E	PD Page Boundary Exceeded	An attempt has been made to update a VF's page descriptor but the request has crossed from one PD page to the next PD page.
0xFFFF	0x800F	SD Boundary Exceeded	Returned by commit FPM (if requested number of HMC objects crosses SD boundary for that function).
0xFFFF	0x8010	Invalid Function Type	The operation is not valid for the PCI function type that attempted to issue the operation.
0xFFFF	0x8011	FLM Not Initialized	Returned by create QP if an attempt to create a QP has been attempted before the static resources allocated WQE has been issued.
0xFFFF	0x8012	Invalid ECN Codepoint	An attempt was made to create a QP with an invalid initial ECN codepoint value.
0xFFFF	0x8013	Bad Access	A PF or VF attempted to access resources that it does not own.
0xFFFF	0x8014	Invalid MRTE Index	An attempt was made to access a reserved STag.
0xFFFF	0x8015	Invalid Terminate Message Size	A Terminate was requested with a *Term_length* value that is too small.
0xFFFF	0x8016	Bad Send MSS Value	The value given for SndMSS is invalid.
0xFFFF	0x8017	Shared MR FBO Mismatch	A Shared Memory Region was created with a virtual address that did not match the First Buffer Offset of the parent Memory Region.
0xFFFF	0x8018	Reserved	Reserved.
0xFFFF	0x8019	Invalid Access Rights	An invalid combination of access rights was requested.
0xFFFF	0x8020	Invalid flag on Reg MR	Memory Region bit was no set during a Register MR operation.
0xFFFF	0x8021	Invalid PBL/Host Page	Unsupported PBL/Host Page Size configuration was requested.
0xFFFF	0x8023	Invalid Push Ready List	An attempt has been made to enable Push Mode with a ready list that was not associated with a valid Push Mode Page Table.
0xFFFF	0x8024	Invalid RoCE QPID	An attempt was made to create a RoCE QP (RC or UD) with an ID of 0 or a RoCE RC QP with an ID of 1.
0xFFFF	0x8025	Invalid MW Type Used	An attempt was made to allocate a Type1 Memory Window on a HMC function not configured for the RoCEv2 protocol.
0xFFFF		Window Type Not Allowed	Type 1 Windows are not valid for iWARP.
0xFFFF	0x80EE	Unsupported Opcode	The opcode given is unsupported.

Table 11-19. CQP Error Codes [continued]

Major Error Code	Minor Error Code	Completion Reason	Description
0xFFFF	0x80EF	WQE Not Valid	The WQE fetched due to a CQP doorbell ring is not valid.
0xFFFF	0xFFFF	Error	CQP encountered an error during the processing of this WQE.

11.5.2.5 CQP CQE Format

The CQP CQ entry is listed in Table 11-20. Every work request processed by the control QP returns a completion on CQ 0 (CQ_ID 0). If CQ0 is created with the *Avoid_Memory_Conflicts* bit set, an additional 32 bytes of padding is added to each CQE for a total of 64 bytes per CQE. This doubles the size of the CQ in host memory.

Table 11-20. CQP CQ Entry Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	RSVD		
8	[63:0]	QP_Completion_Context		
16	[63:32]	RSVD	[31:0]	Operation_Return_Value
24	[63] [62] [61:56] [55]	CQE_Valid SQ Op Error	[54:47] [46:32] [31:16] [15:0]	RSVD WQ_Desc_Index Major_Error_Code Minor_Error_Code

Op (6 bits)

This field reports the opcode from the operation associated with the CQE.

QP_Completion_Context (64 bits)

This field is transferred to the CQE from CQP QP context listed in Table 11-18.

WQ_Desc_Index (15 bits)

WQ are sliced up into 32-byte descriptor quanta. Every WQE must start with a 32-byte descriptor on a 32-byte boundary. *WQ_Desc_Index* reports the 32-byte quanta index of the WQE associated with the completion.

QP_ID (18 bits)

QP associated with the completed message.

Operation_Return_Value (32 bits)

This field is filled in by CQP with operation dependent return values such as the reachability timestamp or the MAC and IP table Index.

CQE_Valid (1 bit)

The *CQE_Valid* bit for CQE indicates that a CQE is ready to be processed. The polarity of the *Valid* bit changes each time the CQ wraps from the last entry back to the first entry. This change in polarity reduces software overhead by avoiding the need to clear the *Valid* bit once software has processed a valid CQE.

Software is responsible to clear (set to 0b) all memory in a CQ initially at CQ creation. The first iteration (and subsequent odd numbered iterations) through the CQ, the E810 sets the *Valid* bit to 1b when it writes a new CQE. For the second iteration (and all even numbered iterations) through the CQ, the E810 sets the *Valid* bit to a 0b when it writes an new CQE.

Error (1 bit)

The *Error* bit indicates if there was a problem with the operation specified by the WQE associated with the WR that is associated with completion.

Value	Description
0b	No error is being reported with this completion.
1b	An error occurred when processing the WQE associated with this CQE and that the *Error_Code* field is valid.

SQ (1 bit)

The SQ bits is used to determine if the completion is associated with a SQ (SQ) or a RQ (RQ). Always 1b for CQP.

Value	Description
0b	RQ
1b	SQ

Major_Error_Code (16 bits) and Minor_Error_Code (16 bits)

The *Error_Code* fields are valid if the *Error* bit is set or if the operation submitted was query STag. See Table 11-19 for the values that are returned for CQP operations.

11.5.3 CQP SQ Descriptor Format

The following WQE formats are used in conjunction with the Control QP (CQP) to manage internal PE structures that the E810 uses to communicate with host software. Operations that are supported for CQPs are the following:

Table 11-21. CQP Operations

Operation Code	Operation Name	Section Reference
0x00	Create QP	11.5.3.2
0x01	Modify QP	
0x02	Destroy QP	
0x03	Create CQ	11.5.3.3
0x04	Modify CQ	
0x05	Destroy CQ	
0x06-0x08	Reserved	N/A
0x09	Allocate STag	11.5.3.4
0x0A	Register MR	
0x0B	Query STag	11.5.3.5
0x0C	Register Shared MR	11.5.3.4
0x0D	Deallocate STag	
0x0E	Reserved	N/A
0x0F	Manage ARP Cache	11.5.3.6
0x10	Manage VF PBLE Backing Pages	11.5.3.7
0x11	Manage Push Page	11.5.3.8
0x12	RDMA Get Features	11.5.3.29
0x13	Upload Context	11.5.3.9
0x14	Reserved	N/A
0x15	Manage HMC PM Function Table	11.5.3.10
0x16	Create CEQ	11.5.3.11
0x17	Reserved	N/A
0x18	Destroy CEQ	11.5.3.11
0x19	Create AEQ	11.5.3.12
0x1A	Reserved	N/A
0x1B	Destroy AEQ	11.5.3.12
0x1C	Create Address Handle	11.5.3.13
0x1D	Modify Address Handle	
0x1E	Destroy Address Handle	
0x1F	Update PE SDs	11.5.3.14
0x20	Query FPM Values	11.5.3.15
0x21	Commit FPM Values	11.5.3.16
0x22	Flush WQEs	11.5.3.17
0x23	Manage APBV Table Entry	11.5.3.18
0x24	NOP	11.5.3.19
0x25	Manage Quad Hash Table Entry	11.5.3.20
0x26	Create Multicast Group	11.5.3.21
0x27	Modify Multicast Group	
0x28	Destroy Multicast Group	
0x29	Suspend QP	11.5.3.22
0x2A	Resume QP	11.5.3.23
0x2B	Static HMC Pages Allocated	11.5.3.24
0x2C	Manage Work Scheduler Node	11.5.3.27
0x2D	Manage Statistics Instance	11.5.3.25
0x2E	Gather Statistics	11.5.3.26
0x2F	Set UP-UP Mapping	11.5.3.28
0x30-0x3F	Reserved	N/A

11.5.3.1 Common CQP Descriptor Format Fields

The basic CQP WQE is a 64-byte structure that is broken up into 64-bit (8-byte) words. The placement of these fields within the WQE are common among all CQP WQEs and also RDMA WQEs. Table 11-22 lists the basic structure of a CQP WQE including the common fields. The definition of fields marked as "Operation Code Dependent" vary from operation-to-operation and are detailed in subsequent sections.

Table 11-22. CQP Common WQE Fields

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	Operation Code Dependent		
8	[63:0]	Operation Code Dependent		
16	[63:0]	Operation Code Dependent		
24	[63] [62:42] [41:38]	WQE_Valid Operation Code Dependent RSVD (AdditionalFragmentCount)	[37:32] [31:0]	OP Operation Code Dependent
32	[63:0]	Operation Code Dependent		
40	[63:0]	Operation Code Dependent		
48	[63:0]	Operation Code Dependent		
56	[63:0]	Operation Code Dependent		

OP (6 bits)

CQP operation code for create QP, modify QP, or destroy QP. See Table 11-21 for the specific values.

WQE_Valid (1 bit)

The *WQE_Valid* bit for WQ Entries (WQE) indicates that a WQE is ready to be processed by the E810. The polarity of the *Valid* bit changes each time the WQ wraps from the last entry back to the first entry. This change in polarity reduces software overhead associated with the need to clear a *Valid* bit and also to enable the E810 to read ahead in the WQ to reduce the need for doorbell rings. See Section 11.4.1.5 for more information on submitting work to a QP with the E810.

Software is responsible to clear (set to 0b) all memory in a WQ initially at QP creation. The first iteration (and subsequent odd numbered iterations) through the WQ, software sets the *Valid* bit to 1b when it writes a new WQE. For the second iteration (and all even numbered iterations) through the WQ, software sets the *Valid* bit to a 0b when it writes an new WQE.

11.5.3.2 Create/Modify/Destroy QP Descriptor Format

This WQE format is used by host software to manage QPs. The various *Valid* bits in this format are designed to reduce the load on CQP for determining what has changed. If a *Valid* bit is set for a particular field, CQP updates the value in the E810's context. Otherwise, it does not. The only valid operation for QP type CQP is the destroy operation. See Section 11.5.2.1 for more details on created CQP.

Note: Software is responsible for populating the appropriate HMC objects and have having created objects for the HMC and through CQP that the QP depends on.

Table 11-23. CQP Create/Modify/Destroy QP WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	RSVD		
8	[63:52] [51:48]	RSVD Term_Length[1]	[47:0]	RSVD
16	[63:0]	QP_Context_Address[1]		
24	[63] [62:60] [59] [58] [57:56] [55] [54] [53:52] [51] [50:48]	WQE_Valid Next_RDMA_State[1] ARP_Table_Index_Valid[1] Reset_Connection[1] Terminate_Actions[1] Remove_Hash_Entry[1] Ignore_MW_Bound[2] RSVD MAC_Valid QP_Type	[47] [46] [45] [44] [43] [42] [41:38] [37:32] [31:18] [17:0]	CQ_Numbers_Valid[1,3] Force_Loopback[3] Virtual_WQs[3] Cached_Variables_Valid[1] TOE_Context_Valid[3] ORD_Valid[1,3] RSVD (AdditionalFragmentCount) OP RSVD QP_ID
32	[63:0]	RSVD		
40	[63:7]	Doorbell_Shadow_Address[3]	[6:0]	RSVD
48	[63:0]	RSVD		
56	[63:0]	RSVD		

1. Not used for Destroy QP operations.
2. Only used for Destroy QP operations.
3. Valid for Create and Modify QP operations as long as the iWARP state is not already RTS, error, or terminate.

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

ARP_Table_Index_Valid (1 bit)

Value	Description
0b	CQP must ignore ARP *Cache_Index* field in QP context.
1b	ARP cache index has been updated in the QP context.

TOE_Context_Valid (1 bit)

Value	Description
0b	Ignore the TOE portion of QP context.
1b	The TOE portion of QP context is valid.

This bit is useful if an iWARP QP is created that does not yet have a TOE connection associated with it. Host software can create the SQ/RQ and so on, and then later issue a modify QP to update the TOE context. Updating TOE context is only allowed prior to setting the QP iWARP state to RTS for iWARP QPs and prior to setting the QP TOE state to established for TOE QPs.

ORD_Valid (1 bit)

Value	Description
0b	CQP must ignore the *ORD* field in QP context.
1b	ORD has been updated in QP context.

Virtual_WQs (1 bit)

Value	Description
0b	The SQ and RQ for this QP are physically mapped.
1b	The SQ and RQ for this QP are virtually mapped.

When a QP is using virtual WQs, the RQ and SQ base address in QP context point to the first HMC FPM space PBLE index for the page table for each WQ. Host software is responsible for populating the PBLEs with the page list associated with each WQ before making the QP active (Modify QP with next state of RTS). Both the SQ and the RQ buffers must be aligned to a multiple of 4 KB in host memory when *Virtual_WQs* is set. This field is only valid for QPs in the IDLE state or during the IDLE to RTS transition.

Remove_Hash_Entry (1 bit)

Only valid with Modify QP opcodes.

Value	Description
0b	No effect.
1b	The hash table entry associated with this QP is removed from the hash list.

CQ_Numbers_Valid (1 bit)

Value	Description
0b	*RxCmpQueueNum* and *TxCmpQueueNum* in QP context are ignored.
1b	The values for *RxCmpQueueNum* and *TxCmpQueueNum* in QP context are used to update internal context.

Force_Loopback (1 bit)

This field is used to enable loopback traffic for different IP Addresses with a single VSI.

Value	Description
0b	The traffic is switched normally unless the source and destination IP Addresses and MAC Addresses match.
1b	The traffic for the QP is internally looped back within the VSI associated with the QP.

QP_Type (3 bits)

QP_Type indicates the type of QP for CQP to create. The valid QP types are:

Value	Description
000b	CQP (only valid for DestroyQP)
001b	iWARP
010b	UDA
011b	RoCEv2 RC
100b	RoCEv2 UD
Note: All other values are reserved.	

MAC_Valid (1 bit)

This field is for changing the MAC after RTS. Before RTS, this field is ignored.

Value	Description
0b	CQP must ignore *Src_MAC_Address* field in QP context.
1b	*Src_MAC_Address* has been updated in QP context.

Ignore_MW_Bound (1 bit)

This field is only meaningful for Destroy QP for RDMA QPs.

Value	Description
0b	The destroy fails with a major/minor code of 0xFFFF/0x4000 when memory windows are still bound to the QP.
1b	The QP is destroyed even if memory windows are still bound to the QP.

Terminate_Actions (2 bits)

Only valid for Modify QP for iWARP QPs with *Next_RDMA_State* set to terminate. Values for this field are:

Value	Description
00b	Send both Terminate and FIN
01b	Send Terminate Only
10b	Send FIN Only
11b	Do not send Terminate or FIN

Term_Length (4 bits)

Only valid with modify opcodes when the next RDMA state is set to 5 (terminate) and *Terminate_Actions* is set to 0b or 1b. This field is ignored in all other cases. This field specifies the number of 4-byte DWords to transfer from the Q2 outbound terminate data area when a terminate message is requested. The Q2 area starts with a DWord specifying the *Terminate Control* field as specified in the RDMAP RFC. A maximum of 13 DWords are transmitted (terminate control DWord plus 48 bytes of data for an RDMA read header). Values greater than 13 results in 13 DWords being transmitted.

Cached_Variables_Valid (1 bit)

Only valid with modify opcodes.

Value	Description
0b	No effect.
1b	Only the cached variables associated with this QP are updated on the Modify operation.

See Table 11-67 on page 1490 for the list of variable that are cached. Each cached variable is marked with a table footnote.

Next_RDMA_State (3 bits)

Only valid on Modify QP opcodes for RDMA QPs. The iWARP state definitions are:

Value	Description
000b	NON EXISTENT
001b	IDLE
010b	RTS
011b	CLOSING
100b	Reserved
101b	TERMINATE (iWARP only)
110b	ERROR
111b	Reserved

Reset_Connection (1 bit)

Only valid with Modify QP opcodes. If set (1b), a TCP reset is sent out if the TCP state for the QP has already been ESTABLISHED is not TIME_WAIT or CLOSED. If the QP TCP state is already TIME_WAIT or CLOSED, an AE_RESET_NOT_SENT AE is generated.

QP_ID (18 bits)

QP_ID identifies the QP that is to be acted upon by the E810.

QP_Context_Address (64 bits)

A physically-mapped pointer in host memory that contains QP context. The format of the QP context structures are define in Section 11.6.2, Section 11.5.2.3, and Section 11.6.3. This buffer must be aligned to a multiple of four bytes.

Doorbell_Shadow_Address (57 bits)

A physically-mapped pointer in host memory that hardware writes to when forward progress has been made on SQ work. The format of the doorbell shadow area is listed in Table 11-25. This buffer must be aligned to a multiple of cache-line size bytes.

Modify QP processing performed by CQP for the E810 is not designed to be verbs compliant from the point of view of which iWARP state transitions are allowed. The E810 enables invalid state transitions to enable host hardware to make state transitions to ERROR or TERMINATE autonomously to prevent subsequent work from being processed, but still enable host software the ability to provide verbs compliant behavior to the application.

Additionally, the E810 is not providing support for the transition back to IDLE. It is expected that host software destroys QPs and recreate them when the transition from non-IDLE to IDLE is desired. Table 11-24 describes the transitions allowed by the E810.

Table 11-24. E810 iWARP QP State Transitions

Current iWARP State	Next RDMA State				
	Idle	RTS	Closing	Terminate	Error
Idle	Yes	Yes	No	No	Yes
RTS	No	Yes	Yes	Yes	Yes
Closing	No	No	No	No	Yes
Terminate	No	No	No	Yes	Yes
Error	No (No error reported)	No	No	No	Yes

Table 11-25. QP Doorbell Shadow Area

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:15]	RSVD	[14:0]	HW_SQ_Tail
8	[63:0]	RSVD		
16	[63:0]	RSVD		
24	[63:0]	RSVD		
32	[63:0]	RSVD		
40	[63:0]	RSVD		
48	[63:0]	RSVD		
56	[63:0]	RSVD		

For information on the doorbell ringing algorithm and usage of the QP doorbell shadow area, see Section 11.4.1.5.2.

11.5.3.3 Create/Modify/Destroy CQ Descriptor Format

Host software uses the following structure to manage CQs through the control QP. CQs are comprised of a packed array of CQEs (which are WQ type specific) that are managed as a circular queue. They can be either virtually or physically contiguous memory buffers. Further discussion of CQ operation is found in Section 11.4.1.3.

Table 11-26. CQP Create/Modify/Destroy CQ WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:21]	RSVD	[20:0]	cq_size[1]
8	[63]	RSVD	[62:0]	CQ_Context_Value
16	[63:18]	RSVD	[17:0]	cq_shadow_read_threshold[2]
24	[63] [62] [61] [60] [59:50] [49] [48] [47]	WQE_Valid RSVD Avoid_Memory_Conflicts[2] TPH_en RSVD CEQ_ID_Valid enable_ceqe_mask[2] Virtually_Mapped[1]	[46] [45:44] [43] [42:38] [37:32] [31:22] [21:19] [18:0]	Check_Overflow[2] Leaf_PBL_Size[1] CQ_Resize RSVD OP CEQ_ID RSVD CQ_ID
32	[63:8]	Physical_Buffer_Address[1]	[7:0]	RSVD
40	[63:6]	Doorbell_Shadow_Address	[5:0]	RSVD
48	[63:28]	RSVD	[27:0]	first_pm_pbl_index[1]
56	[63:18] [17:8]	RSVD VSI_Index	[7:0]	TPH_Value

1. Valid on Modify operations with the *CQ_Resize* bit set, or on Create operations.
2. Only valid on Create operations.

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

enable_ceqe_mask (1 bit)

Value	Description
0b	The E810 potentially generates multiple CEQEs per CQ if a request for completion notification is generated by software before a CEQE is consumed by software. This behavior can lead to CEQ overflows.
1b	The E810 only allows one CEQE to be generated per CQ. Subsequent CEQE generation is enabled by writing the CQ ACK register.

CEQ_ID (10 bits)

CEQ_ID specifies the CEQ to associated with the CQ specified by *CQ_ID*. This field is only meaningful when *CEQ_ID_Valid* is set (1b).

CEQ_ID_Valid (1 bit)

Value	Description
0b	The *CEQ_ID* field is ignored.
1b	The E810 updates the CEQ for the CQ.

Changing the *CEQ_ID* for an active CQ is allowed to enable software to redistribute CQ event processing between different processors. Each CEQ for a given function must be assigned to independent MSI-X vectors to take advantage of this capability.

CQ_Resize (1 bit)

Value	Description
0b	Ignore *CQ_Resize*.
1b	CQ resize operation is being requested.

If a resize operation is performed, the *cq_resize_count* CQ context variable (known as *hw_cq_select* in the description of CQ resizing) is incremented by the E810. See the description of CQ resizing in Section 11.4.1.3 for more information on the CQ resizing operation.

Avoid_Memory_Conflicts (1 bit)

Value	Description
0b	Each CQE is 32 bytes in size.
1b	The size of each CQE is padded with 0's to a total of 64 bytes, and the size of the CQ doubles.

TPH_en (1 bit)

Value	Description
0b	TPH is not used for this resource.
1b	TPH is enabled for this resource.

TPH_Value (8 bits)

If *TPH_en* is set (1b), TPH STag is initialized with *TPH_Value*. If *TPH_en* is clear (0b), this field is ignored.

VSI_Index (10 bits)

The VSI index for the CQ. Only defined for Create CQ. It is ignored on Modify CQ and Destroy CQ.

Leaf_PBL_Size (2 bits)

See Section 11.4.1.4.1 for more details on usage of two level PBLs.

Value	Description
00b	Reserved
01b	Variable (one level)
10b	256 bytes (two level)
11b	4 KB (two level)

A leaf page size of 4 KB is not supported for 1 GB pages.

Check_Overflow (1 bit)

Value	Description
0b	It is host software's responsibility to ensure that a CQ cannot overflow.
1b	The E810 checks for CQ overflow conditions.

Virtually_Mapped (1 bit)

Value	Description
0b	*Physical_Buffer_Address* is the physical address of the physically-contiguous CQ ring buffer.
1b	The CQ is virtually-mapped, and *first_pm_pbl_index* is the index of the PBLE HMC object to use for mapping the virtually-contiguous CQ.

The CQ buffer must be aligned to a multiple of 4 KB in host memory when *Virtually_Mapped* is set.

first_pm_pbl_index (28 bits)

Points to the first HMC FPM space PBLE index for the page table for the CQ. Host software is responsible for populating the PBLEs with the page list associated with the CQ before issuing the create CQ operation and the page table is not allowed to be changed.

Physical_Buffer_Address (56 bits)

If *Virtually_Mapped* is set to 0b, this field specifies the physical buffer address associated with the queue in host memory. If *Virtually_Mapped* is set to 1b, this field is ignored by the E810. This address must be aligned to a 256-byte boundary.

cq_size (21 bits)

The number of CQEs allowed on the CQ. Note that the actual number of concurrent CQEs that might be allocated to the CQ is *cq_size* -1. 0 and 1 are invalid sizes for a CQ. Also note that each Immediate Data operation CQE consumes the space of two CQEs.

cq_shadow_read_threshold (18 bits)

Controls when the CQ shadow area is read to update hardware's copy of *CQE_Index*. When the CQ drops below the number for CQEs specified by *cq_shadow_read_threshold*, the CQ shadow area is read to determine which CQEs have been processed by software. The value of 0 indicates that the CQ shadow area is only read when the CQ has no more available entries to use for a new CQE that needs to be generated.

CQ_ID (19 bits)

Identifies the CQ index associated with the current CQP operation.

CQ_Context_Value (63 bits)

This value is returned in the CEQE when an event is generated for the CQ specified by *CQ_ID*. This value is intended to be used by host software to quickly locate host software's CQ context (virtual pointer to CQ context).

Doorbell_Shadow_Address (58 bits)

A physically-mapped pointer in host memory that hardware reads to determine the CQEs that are owned by hardware. The format of the doorbell shadow area is listed in Table 11-27. This buffer must be aligned to a multiple of cache-line size bytes or 64 bytes, whichever is greater.

Table 11-27. CQ Doorbell Shadow Area

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:21]	RSVD	[20:0]	CQE_Index
8	[63:0]	RSVD		
16	[63:0]	RSVD		
24	[63:0]	RSVD		
32	[63:18] [17:16] [15]	RSVD arm_seq_num arm_next_se	[14] [13:0]	arm_next sw_cq_select
40	[63:0]	RSVD		
48	[63:0]	RSVD		
56	[63:0]	RSVD		

CQE_Index (21 bits)

Indicates the next CQE that software are polling through verbs. This field is used to update the hardware's version of context to detect CQ overrun conditions.

arm_next (1 bit)

Indicates that software wants a CEQE generated for the next CQ entry posted to the CQ.

arm_next_se (1 bit)

Indicates that software wants a CEQE generated for the next CQ entry posted to the CQ that is associated with a solicited event operation.

arm_seq_num (2 bits)

Arm Sequence Number (*arm_seq_num*) is a valued incremented by software each time an arm request is issued. It is expected that no more than two arm requests are issued per CE received. If more than three arm requests are issued without waiting for a CE, arm request might be lost.

sw_cq_select (14 bits)

This field is used for CQ resize operations. See Section 11.4.1.3 for more details on the usage of this field.

11.5.3.4 Allocate/Register/Registershared/Deallocate STag Descriptor Format

This WQE format is used by host software to manage memory regions and windows. See Section 11.4.1.4.1 for more details on memory registration concepts with the E810.

Table 11-28. CQP Allocate/Register/Registershared/Deallocate STag WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	Virtual_Address or First_Buffer_Offset		
8	[63:46]	PD_ID	[45:0]	STag_Length
16	[63:54] [53:32]	RSVD Parent_STag_Index	[31:8] [7:0]	Driver_Key_STag_Index Consumer_Key
24	[63] [62:61] [60] [59] [58] [57:54] [53] [52:48]	WQE_Valid RSVD use_hmc_fcn_index VA_Based_TO MW1_bind_dont_vldt_key RSVD (for PM flags) Remote_Access_Enabled Access_Rights	[47:46] [45:44] [43] [42] [41:38] [37:32] [31:0]	Host_Page_Size Leaf_PBL_Size Memory_Region Memory_Window_Type RSVD OP RSVD
32	[63:0]	Physical_Buffer_Address		
40	[63:6]	RSVD	[5:0]	hmc_fcn_index
48	[63:28]	RSVD	[27:0]	first_pm_pbl_index
56	[63:0]	RSVD		

OP (6 bits) and WQE_Valid (1 bits)

See Section 11.5.3.1.

Virtual_Address or First_Buffer_Offset (64 bits)

Indicates the base VA for this region/window for VA-based entries and indicates the first buffer offset for zero-based entries. For VA-based entries, the least significant bits (12 bits for entries based on 4 KB pages, 21 bits for entries based on 2 MB pages) of the VA indicate the offset in bytes from the beginning of the first page where the entry begins.

Physical_Buffer_Address (64 bits)

If *Leaf_PBL_Size* is set to zero (00b), this field specifies the physical buffer address associated with the virtual address. If *Leaf_PBL_Size* is not set to zero, this field is ignored by the E810. This address must be aligned to the page size specified in *Host_Page_Size*. Any offset into the page is ignored by the E810.

PD_ID (18 bits)

Indicates the protection domain ID associated with the memory region or window.

STag_Length (46 bits)

Length of the memory region or memory windows specified by the STag index specified by *Driver_Key_Stag_Index*. A value of 0b indicates that the length is not checked. This is typically used for systems that want to address all of RAM with a single STag. The maximum size that can be registered with a single STag with 4 KB pages and 1-level PBLs is 1 TB (2^{28} 4 KB pages). The maximum size that can be registered with a single STag with 2 MB pages and 1-level PBLs is 32 TB (2^{24} 2 MB pages). The maximum size that can be registered with a single STag with 1 GB pages and 1-level PBLs is 32 TB (2^{15} 1 GB pages).

Parent_STag_Index (22 bits)

Index of the STag associated with the memory window (*Memory_Region* = 0) or linked memory region (*OP* = Register Shared MR).

Driver_Key_STag_Index (24 bits)

Index and *Driver Key* fields of the STag associated with the memory window (*Memory_Region* = 0b) or linked memory region (*OP* = register shared MR). The E810 supports a variable size STag index. This means that the number of bits used for *Driver Key* and for *STag_Index* are dependent on the maximum number of STag supported for a given PCI function. For example, if a PCI function read the FPMPEMRSZ field (See Section 13.2.2.20.130) and found that the maximum number of MRTEs was 64 KB, the lower 16 bits of this field would be the STag Index and the upper eight bits would be a driver key that the driver can randomize to make guessing the MRTE layout more difficult to guess. STag index 4M-1 (4,194,303) is reserved for hardware use. The completion returns an error if this reserved STag index is specified.

The width of the *STag_Index* field is based on two things: the Memory Region count supplied in the Commit FPM Buffer and the bit width mask size defined in the MRTE Index Mask (PFPE_MRTEIDXMASK) register. If the Memory Region count requires fewer bits that the minimum mask size, the STag Index size is adjusted up to the minimum. The driver must not randomize bits in the range of the STag index mask bits.

Consumer_Key (8 bits)

Consumer key is the least significant 8-bit portion of the STag. This field is supplied by the user application or the driver.

use_hmc_fcn_index (1 bit)

Only valid when issued to a PF CQP instance. Ignored (treated as if set to 0b) for VF CQP instances.

Value	Description
0b	The HMC function index is determined by the PCI function associated with the PCI function that issues the CQP operation.
1b	*hmc_fcn_index* can be set to a HMC function index that is a VF associated with the PF associated with the CQP instance used to issue the command.

hmc_fcn_index (6 bits)

Only valid when issued to a PF CQP instance and *use_hmc_fcn_index* is set to 1b. Ignored otherwise.

VA_Based_TO (1 bit)

VA_Based_TO specifies if the STag is zero-based or Virtual Address (VA)-based. Zero-based STags carry only the first buffer offset in the VA or First Buffer Offset field. VA-based STags carry the full base VA including first buffer offset in the VA or *First_Buffer_Offset* field. The STag is VA-based if *VA_Based_TO* is set (1b). Otherwise, the STag is zero-based.

Remote_Access_Enabled (1 bit)

If set, the entry is enabled for remote access. See the remote access flag in verbs for more details.

Access_Rights (5 bits)

Indicates the rights assigned to this STag. The values for this field are:

Value	Description
00001b	Enable local read
00010b	Enable local write
00100b	Enable remote read
01000b	Enable remote write
10000b	Enable window bind
Note: All other values are reserved.	

Host_Page_Size (2 bits)

Host_Page_Size specifies the page size of the backing pages for the STag. The values for this field are:

Value	Description
00b	4 KB pages
01b	2 MB pages
10b	1 GB pages
11b	Reserved

Leaf_PBL_Size (2 bits)

The E810 supports physically-contiguous STags and two forms of virtually-contiguous STags. Physically-contiguous STag do not require any PBLs and store physical address of the first page of the STag directly with the STag (No leaf PBL). Virtually-contiguous STags that can be represented with a single HMC virtually-contiguous address range require a single level PBL of Variable size. Virtually-contiguous STags that are large (or in cases where the HMC address space for PBLs becomes fragmented) might require two level PBLs. In this case, the E810 needs to know the length of the leaf PBLs to properly manage access to the PBLs. The valid settings for *Leaf_PBL_Size* are the following:

Value	Description
00b	No leaf PBL
01b	Variable (one level)
10b	256 bytes (two level)
11b	4 KB (two level)

Memory_Region (1 bit)

Value	Description
0b	The entry describes a memory window.
1b	The entry describes a memory region.

Memory_Window_Type (1 bit)

Valid only if *Memory_Region* is clear (0b) indicating a memory window.

Value	Description
0b	The entry describes a Type 2B memory window.
1b	The entry describes a Type 1 memory window (sometimes called a wide window).

Note: Type 1 Windows are not valid for iWARP.

MW1_bind_dont_vldt_key (1 bit)

Specifies whether the consumer key is validated on a bind of a Type 1 memory window. This bit is valid only if this is an Allocate of a Type 1 Memory Window.

Value	Description
0b	The consumer key is validated before being updated.
1b	The consumer key is not validated before being updated.

first_pm_pbl_index (28 bits)

Designates the HMC base address for the PBLs for this STag.

Table 11-29 shows the WQE fields that are valid for the STag management WQEs. Fields that are marked as not valid are ignored by the E810.

Table 11-29. WQE Fields Valid for Allocate/Register/Registershared/Deallocate STAG

WQE Field	Allocate	Register	Register Shared	Deallocate	Comments
Memory_Region	Valid	Valid	Not Valid	Valid	
VA_Based_TO	Not Valid	Valid	Valid	Not Valid	
Leaf_PBL_Size	Valid if *MR*=1	Valid	Not Valid	Not Valid	
Access_Rights	Not Valid	Valid	Valid	Not Valid	
Remote_Access_Enabled	Valid	Valid	Valid	Not Valid	
PD_ID	Valid	Valid	Valid	Valid	
Driver_Key_STAG_Index	Valid	Valid	Valid	Valid	
Parent_STAG_Index	Not Valid	Not Valid	Valid	Not Valid	
Consumer_Key	Not Valid	Valid	Valid	Not Valid	Host software is expected to validate the consumer key on a deallocate and on register shared.
*Base_VA*_or_*First_Buffer_Offset*	Not Valid	Valid	Valid if *VA_Based_TO*=1	Not Valid	
Physical_Buffer_Address	Not Valid	Valid	Not Valid	Not Valid	
first_pm_pbl_index	Valid	Valid	Not Valid	Not Valid	
STag_Length	Valid if *MR*=1	Valid	Not Valid	Not Valid	
Host_Page_Size	Valid	Valid	Not Valid	Not Valid	
use_hmc_fcn_index	Valid	Valid	Valid	Valid	This field is ignored (assumed to be 0b) for VFs.

Table 11-29. WQE Fields Valid for Allocate/Register/Registershared/Deallocate STAG

WQE Field	Allocate	Register	Register Shared	Deallocate	Comments
hmc_fcn_index	Valid	Valid	Valid	Valid	Only used when *use_hmc_fcn_index* is 1b.
Memory_Window_Type	Valid	Not Valid	Not Valid	Not Valid	

11.5.3.5 Query STag Descriptor Format

WQE returns the STag value and current state of an STag (valid or invalid) and the consumer key in the completion for the WQE. See Table 11-19 for information on CQP completion codes that indicate the STag state. The full STag value is returned in the *Operation_Return_Value*. Note that the error bit in the CQE should be ignored for query STag operations.

Table 11-30. CQP Query STag WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	RSVD		
8	[63:0]	RSVD		
16	[63:30] [29:8]	RSVD STag_Index	[7:0]	RSVD
24	[63] [62:61] [60] [59:42]	WQE_Valid RSVD use_hmc_fcn_index[1] RSVD	[41:38] [37:32] [31:0]	RSVD (AdditionalFragmentCount) OP RSVD
32	[63:0]	RSVD		
40	[63:6]	RSVD	[5:0]	hmc_fcn_index[1]
48	[63:0]	RSVD		
56	[63:0]	RSVD		

1. This field is only valid for PF operations and is treated as 0b for VFs.

OP (6 bits) and WQE_Valid (1 bits)

See Section 11.5.3.1.

STag_Index (22 bits)

STag_Index specifies the index of the STag to be queried.

use_hmc_fcn_index (1 bit)

Only valid when issued to a PF CQP instance. Ignored (treated as if set to 0b) for VF CQP instances.

Value	Description
0b	The HMC function index is determined by the PCI function associated with the PCI function that issues the CQP operation.
1b	*hmc_fcn_index* can be set to a HMC function index that is a VF associated with the PF associated with the CQP instance used to issue the command.

hmc_fcn_index (6 bits)

Only valid when issued to a PF CQP instance and *use_hmc_fcn_index* is set to 1b. Ignored otherwise.

11.5.3.6 Manage ARP Table Descriptor Format

This WQE format is used to manage the E810's ARP table. An ARP table entry must be created before a QP can be transitioned to RTS or established since the QP context references an ARP table entry.

Table 11-31. Manage ARP Table WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	RSVD		
8	[63:32]	RSVD	[31:0]	Reachability_Max
16	[63:48]	RSVD	[47:0]	MAC_Address
24	[63] [62:45] [44] [43] [42}	WQE_Valid RSVD Query Permanent Entry_Valid	[41:38] [37:32] [31:16] [15:0]	RSVD (AdditionalFragmentCount) OP RSVD ARP_Entry_Index
32	[63:0]	RSVD		
40	[63:0]	RSVD		
48	[63:0]	RSVD		
56	[63:0]	RSVD		

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

ARP_Entry_Index (16 bits)

ARP_Entry_Index specifies the index of the ARP table entry to be manipulated.

Permanent (1 bit)

Value	Description
0b	The entry is created in the reachable state and is aged.
1b	The ARP entry is marked as permanent and is not aged.

Query (1 bit)

Value	Description
0b	The ARP entry is written according to the specified parameters.
1b	The ARP entry is not written. Instead, the entry is read and the reachability timestamp from the ARP entry is returned in the *Operation_Return_Value* field of the CQE.

Entry_Valid (1 bit)

Value	Description
0b	The ARP entry is set to the invalid state.
1b	The ARP entry is set to the valid state.

Note: The lower 32 bits of the AE generated when an ARP entry is stale contain the timestamp value from the ARP table entry.

Reachability_Max (32 bits)

The maximum number of microseconds that should be allowed to expire before generating a doubt neighbor reachability AEQE. Doubt neighbor reachability AEs indicate that there has been TCP/IP transmits outstanding for *Reachability_Max* without receiving any inbound traffic from the associated neighbor. To receive subsequent doubt neighbor reachability AEs, a manage ARP table WQE must be re-submitted to CQP with query clear (0b) and all original information supplied when the ARP entry was last updated.

MAC_Address (48 bits)

MAC Address to be placed in the ARP cache table.

11.5.3.7 Manage VF PBLE Backing Pages Descriptor Format

The Manage VF PBLE Backing Pages Descriptor is used to populate and depopulate VF related HMC page descriptor content. A VF that allocates the PBLE HMC object backing pages can request CQP to copy the page list associated with the backing pages to the PF driver allocated page descriptor pages. A VF might also request CQP to invalidate VF related page descriptor entries. This operation reduces the number of VF to PF messages required to populate the VF related HMC PBLE object backing pages.

Table 11-32. CQP Manage VF PBLE Backing Pages WQE Format

Byte Offset	[Bit Range] Field Name			
0	[63:0]	RSVD		
8	[63:0]	RSVD		
16	[63:44] RSVD [43:32] SD_Index [31:25] RSVD		[24:16] First_PD_Index [15:10] RSVD [9:0] PD_Entry_Count	
24	[63] WQE_Valid [62] Invalidate_PD_Entries [61:42] RSVD		[41:38] RSVD (AdditionalFragmentCount) [37:32] OP [31:0] RSVD	
32	[63:3]	PD_Pagelist_Physical_Buffer_Address	[2:0]	RSVD
40	[63:0]	RSVD		
48	[63:0]	RSVD		
56	[63:0]	RSVD		

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

Invalidate_PD_Entries (1 bit)

Value	Description
0b	VF PD entries are copied from VF address space to the PF allocated PDs.
1b	The PF allocated PDs are invalidated.

PD_Entry_Count (10 bits)

This field indicates the number of PD entries to be populated or invalidated. Valid values for this field are 1 through 512. Other values generate completion errors for the CQP operation. *PD_Entry_Count* + *First_PD_Index* must not exceed 512 or a completion error is generated.

First_PD_Index (9 bits)

This field indicates the starting index of the PD page. *SD_Index* identifies the PD page, and *First_PD_Index* provides the offset into the PD page that is the destination of the CQP operation. If the PD entries associated with the *SD_Index* and *First_PD_Index* are not related to a PBLE object, a completion error is generated by CQP for this operation.

SD_Index (12 bits)

This field indicates the SD index that identifies the PD page. The SD entries are in no way modified. If the SD is not related to a PBLE object, then CQP returns a completion error.

PD_Pagelist_Physical_Buffer_Address (61 bits)

If *Invalidate_PD_Entries* is clear, this field indicates the 8-byte aligned physical address of the packing page list to be copied to the PD page. This address is a guest physical address. Host software is responsible to properly set the fields of the PD page list entries previous to issuing the request to CQP.

11.5.3.8 Manage Push Page Descriptor Format

The Manage Push Page Descriptor is used to assign a push page from BAR0 to a VSI and TC. For an allocate operation, the push page index assigned by CQP is returned in the *Operation_Return_Value* of the CQE associated with the WQE. For VFs, the push page index is function relative. When GLPE_PSHCFG.*PSHCFG_DB_SPLIT* is set, two pages (one for doorbells and one for push WQEs) are represented by push page index instead of one.

Table 11-33. CQP Manage Push Page WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	RSVD		
8	[63:0]	RSVD		
16	[63:10]	RSVD	[9:0]	QS_Handle
24	[63] [62] [61:60] [59:42]	WQE_Valid Free_Page Push_Page_Type RSVD	[41:38] [37:32] [31:10] [9:0]	RSVD (AdditionalFragmentCount) OP RSVD Push_Page_Index
32	[63:0]	RSVD		
40	[63:0]	RSVD		
48	[63:0]	RSVD		
56	[63:0]	RSVD		

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

Free_Page (1 bit)

Value	Description
0b	A push page is allocated for the VSI and TC specified by *QS_Handle*.
1b	The push page specified by *Push_Page_Index* is freed.

Push_Page_Index (10 bits)

Only valid when *Free_Page* is set (1b). Identifies the push page to be freed. For VFs, the push page index is function relative.

QS_Handle (10 bits)

Identifies the QS handle associated with the push page. See Section 8.3.3.4 for more information on scheduler configuration.

Push_Page_Type (2 bits)

Value	Description
00b	Push page.
01b	Userspace DB alias page (experimental).
Note: All other values are reserved.	

The userspace DB alias page is a way to re-map the doorbell page intended for containers.

11.5.3.9 Upload Context Descriptor Format

Upload Context is used by host software to freeze a QPs state and upload the context to the host.

Table 11-34. CQP Upload Context WQE Format

Byte Offset	[Bit Range] Field Name			
0	[63:0]	RSVD		
8	[63:0]	RSVD		
16	[63:0]	QP_Context_Address		
24	[63] [62] [61:51] [50:48] [47:42]	WQE_Valid Freeze_QP RSVD QP_Type RSVD	[41:38] [37:32] [31:18] [17:0]	RSVD (AdditionalFragmentCount) OP RSVD QP_ID
32	[63:0]	RSVD		
40	[63:0]	RSVD		
48	[63:0]	RSVD		
56	[63:0]	RSVD		

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

QP_ID (18 bits)

Identifies the QP number be uploaded to host memory.

QP_Type (3 bits)

Value	Description
001b	iWARP
010b	UDA

Value	Description
011b	RoCEv2 RC
100b	RoCEv2 UD
Note: All other values are reserved.	

Freeze_QP (1 bit)

Value	Description
0b	The QP continues processing normally after the snapshot of the context has been returned.
1b	The QP is frozen before the context is retrieved.

QP_Context_Address (64 bits)

A physically-mapped pointer in host memory that contains uploaded context structure format listed in Table 11-35. This buffer must be aligned to a multiple of four bytes.

Table 11-35. Uploaded Context Structure Format (iWARP/UDA)

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:56] [55:48] [47:8]	RSVD kalive_timer_probes RSVD	[7:4] [3:0]	RDMA_state TCP_state
8	[63:32]	timestamp_page	[31:0]	timestamp_recent
16	[63:32]	snd_wnd	[31:0]	snd_nxt
24	[63:32]	rcv_wnd	[31:0]	rcv_nxt
32	[63:32]	snd_una	[31:0]	snd_max
40	[63:32]	rtt_var	[31:0]	srtt
48	[63:32]	cwnd	[31:0]	ss_thresh
56	[63:32]	snd_wl2	[31:0]	snd_wl1
64	[63:54] [53:48]	RSVD rexmit_count	[47:32] [31:0]	RSVD max_snd_window
72	[63:47] [46:32] [31] [30:27]	RSVD hw_sq_tail q1_wa RSVD	[26:24] [23:16] [15:0]	dupacks probe_cnt RSVD
80	[63:14]	RSVD	[13:0]	hw_rq_tail
88	[63:0]	RSVD		
96	[63:0]	RSVD		
104	[63:0]	RSVD		
112	[63:0]	RSVD		
120	[63:0]	RSVD		

hw_sq_tail

This is the 32-byte WQE quanta index of the first unacknowledged WQE for the SQ. There is not a CQE generated for this WQE.

hw_rq_tail

This is the 32-byte WQE quanta index of the first incomplete WQE for the RQ. There is not a CQE generated for this WQE.

q1_wa

This value is set (1b) if Q1 (or inbound RDMA read queue) work is pending when the context was uploaded. The value is clear (0b) if no Q1 work was pending when the context was uploaded.

kalive_timer_probes

The number of retransmits that have been sent (RFC 1122).

Note: Note: Refer to Section 11.6.2 for more on context variable definitions.

Table 11-36. Uploaded Context Structure Format (RoCEv2)

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:21] [20:16] [15:8]	RSVD ack_credits RSVD	[7:6] [5:3] [2:0]	dctcp_state RoceStateRx RoceStateTx
8	[63:60] [59:32]	RSVD Xmit_tail	[31:28] [27:0]	RSVD Q1_tail
16	[63:60] [59:32]	RSVD rdrresp_tail	[31:0]	RSVD
24	[63:56] [39:32]	RSVD psn_una	[31:24] [23:0]	RSVD ssn_una
32	[63:56] [55:32]	RSVD psn_nxt	[31:0]	RSVD
40	[63:32]	rtt_var	[31:0]	srtt
48	[63:32]	cwnd	[31:0]	RSVD
56	[63:0]	RSVD		
64	[63:54] [53:48]	RSVD rexmit_count	[47:0]	RSVD
72	[63:47] [46:32] [31]	RSVD SQ_tail SQ_WA	[30:24] [23:16] [15:0]	RSVD probe_cnt RSVD
80	[63:14]	RSVD	[13:0]	RQ_tail
88	[63:0]	RSVD		
96	[63:0]	RSVD		
104	[63:0]	RSVD		
112	[63:0]	RSVD		
120	[63:0]	RSVD		

11.5.3.10 Manage HMC PM Function Table

The Manage HMC PM Function Table is used to allocate and free HMC private memory functions for VFs associated with a PF. This WQE is only supported for PF CQPs. For allocate operations (*Free_PM_FCN*=0b), the HMC function index is reported in the *Operation_Return_Value* field of the CQE associated with this WQE if the operation is successful.

Table 11-37. CQP Manage HMC PM Function Table WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	RSVD		
8	[63:0]	RSVD		
16	[63:0]	RSVD		
24	[63] [62] [61:38]	WQE_Valid Free_PM_FCN RSVD	[37:32] [31:8] [7:0]	OP RSVD VF_Index
32	[63:0]	RSVD		
40	[63:0]	RSVD		
48	[63:0]	RSVD		
56	[63:0]	RSVD		

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

Free_PM_FCN (1 bit)

Value	Description
0b	The E810 attempts to allocate a PM function for the VF specified in *VF_Index*.
1b	The E810 frees the PM function specified in *VF_index*.

VF_Index (8 bits)

VF_Index indicates the PCI VF index to be used for the allocation or free request.

11.5.3.11 Create/Destroy CEQ Descriptor Format

Host software uses the following structure to manage CEQs through the control QP. CEQs are comprised of a packed array of CEQ entries (see Section 11.4.5) that are managed as a circular queue. They can be either virtually- or physically-contiguous memory buffers. When MSI-X is enabled, each CEQ might be assigned to independent MSI-X vectors to enable distribution of completion event processing across multiple CPUs. Application's process and MSI-X vector assignment capabilities and mechanisms are specific to each operating system.

Table 11-38. CQP Create/Destroy CEQ WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	RSVD		
8	[63:0]	RSVD		
16	[63:18]	RSVD	[17:0]	ceq_size
24	[63] [62:61] [60] [59:48] [47] [46]	WQE_Valid RSVD TPH_en RSVD Virtually_Mapped itr_no_expire	[45:44] [43:42] [41:38] [37:32] [31:10] [9:0]	Leaf_PBL_Size RSVD RSVD (AdditionalFragmentCount) OP RSVD CEQ_ID
32	[63:8]	Physical_Buffer_Address	[7:0]	RSVD
40	[63:0]	RSVD		
48	[63:28]	RSVD	[27:0]	first_pm_pbl_index
56	[63:18] [17:8]	RSVD VSI_Index	[7:0]	TPH_Value

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

CEQ_ID (10 bits)

CEQ_ID specifies the CEQ to manipulated.

TPH_en (1 bit)

Value	Description
0b	THP is not used for this resource.
1b	TPH is enabled for this resource.

TPH_Value (8 bits)

If *TPH_en* is set (1b), TPH STag is initialized with *TPH_Value*. If *TPH_en* is clear (0b), this field is ignored.

VSI_Index (10 bits)

The VSI index for the CEQ. Only defined for Create CEQ. It is ignored on Destroy CEQ.

first_pm_pbl_index (28 bits)

Points to the first HMC FPM space PBLE index for the page table. Host software is responsible for populating the PBLEs with the page list associated with the object before using the object.

Leaf_PBL_Size (2 bits)

See Section 11.4.1.4.1 for more details on usage of two level PBLs.

Value	Description
00b	Reserved
01b	Variable (one level)
10b	256 bytes (two level)
11b	4 KB (two level)

Virtually_Mapped (1 bit)

Value	Description
0b	*Physical_Buffer_Address* is the physical address of the physically-contiguous CEQ ring buffer.
1b	The CEQ is virtually-mapped, and *first_pm_pbl_index* is the index of the PBLE HMC object of the PBL to use for mapping the virtually-contiguous CEQ.

The CEQ buffer must be aligned to a multiple of 4 KB in host memory when virtually mapped is set.

itr_no_expire (1 bit)

Value	Description
0b	There is a short delay before a CEQE is written and the interrupt is issued (if enabled).
1b	The CEQE is written and the interrupt (if enabled) occurs immediately.

Physical_Buffer_Address (56 bits)

If *Virtually_Mapped* is set to 0b, this field specifies the physical buffer address associated with the queue in host memory. If *Virtually_Mapped* is set to 1b, this field is ignored by the E810. This address must be aligned to a 256-byte boundary.

ceq_size (18 bits)

The number of CEQEs allowed on the CEQ. Note that the actual number of concurrent CEQEs that can be allocated to the CEQ is *ceq_size* -1. 0 and 1 are invalid sizes for a CEQ.

11.5.3.12 Create/Destroy AEQ Descriptor Format

Host software uses the following structure to manage AEQs through the control QP. AEQs are comprised of a packed array of AEQ entries (see Section 11.4.6) that are managed as a circular queue. They can be either virtually- or physically-contiguous memory buffers.

Table 11-39. CQP Create/Destroy AEQ WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	RSVD		
8	[63:0]	RSVD		
16	[63:19]	RSVD	[18:0]	AEQE_Count
24	[63] [62:48] [47] [46] [45:44]	WQE_Valid RSVD Virtually_Mapped RSVD Leaf_PBL_Size	[43:42] [41:38] [37:32] [31:0]	RSVD RSVD (AdditionalFragmentCount) OP RSVD
32	[63:8]	Physical_Buffer_Address	[7:0]	RSVD
40	[63:0]	RSVD		
48	[63:28]	RSVD	[27:0]	first_pm_pbl_index
56	[63:0]	RSVD		

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

first_pm_pbl_index (28 bits)

Points to the first HMC FPM space PBLE index for the page table. Host software is responsible for populating the PBLEs with the page list associated with the object before using the object.

Leaf_PBL_Size (2 bits)

See Section 11.4.1.4.1 for more details on usage of two level PBLs.

Value	Description
00b	Reserved
01b	Variable (one level)
10b	256 bytes (two level)
11b	4 KB (two level)

Virtually_Mapped (1 bit)

Value	Description
0b	*Physical_Buffer_Address* is the physical address of the physically-contiguous AEQ ring buffer.
1b	The AEQ is virtually-mapped, and *first_pm_pbl_index* is the index of the PBLE HMC object of the PBL to use for mapping the virtually-contiguous AEQ.

The AEQ buffer must be aligned to a multiple of 4 KB in host memory when virtually mapped is set.

Physical_Buffer_Address (56 bits)

If *Virtually_Mapped* is set to 0b, this field specifies the physical buffer address associated with the queue in host memory. If *Virtually_Mapped* is set to 1b, this field is ignored by the E810. This address must be aligned to a 256-byte boundary.

AEQE_Count (19 bits)

The number of AEQEs allowed on the AEQ. Note that the actual number of concurrent AEQEs that can be allocated to the AEQ is *AEQE_Count* -1. 0 and 1 are invalid sizes for an AEQ.

11.5.3.13 Create/Modify/Destroy Address Handle Descriptor Format

Host software uses following structure to create, modify or destroy address handle. Address Handle is used by UDA and RoCEv2 UD traffic to allow protected Ethernet and IP Headers generation by hardware, while ULP headers (such as UDP) and datagram payload is provided directly by the application or the software library running in the application address space. Address Handle in conjunction with UDA Queue context carries all information necessary for the header generation.

Internal switching capabilities within the same VSI are controlled using the *DoLoopback* bit in Address Handle. Physical Function driver might need to Modify Address Handle to set or clear this bit at runtime using the Modify Address Handle CQP command, depending on the changes in configuration. For example, UDA and OS consumers subscribed or unsubscribed from the multicast group.

Table 11-40. CQP Create/Modify/Destroy Address Handle WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:56] [55:48] [47:40] [39:32]	Src_MAC_Address[5] Src_MAC_Address[4] Src_MAC_Address[3] Src_MAC_Address[2]	[31:24] [23:16] [15:0]	Src_MAC_Address[1] Src_MAC_Address[0] RSVD
8	[63:48] [47:40] [39:32]	PD_Index RSVD Traffic_Class_or_TOS	[31:16] [15:0]	RSVD VLAN_Tag
16	[63:48] [47:40] [39:32]	ARP_Index RSVD Hop_Limit_or_TTL	[31:22] [21:20] [19:0]	RSVD pd_indexd_high Flow_Label
24	[63] [62] [61] [60] [59]	WQE_Valid DoLoopback RSVD Insert_VLAN_Tag IPv4_Valid	[58:42] [41:38] [37:32] [31:17] [16:0]	RSVD RSVD (AdditionalFragmentCount) OP RSVD AH_ID
32	[63:32]	Dest_IP_Address_2	[31:0]	Dest_IP_Address_3
40	[63:32]	Dest_IP_Address_0	[31:0]	Dest_IP_Address_1
48	[63:32]	Src_IP_Address_2	[31:0]	Src_IP_Address_3
56	[63:32]	Src_IP_Address_0	[31:0]	Src_IP_Address_1

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

IPv4_Valid (1 bit)

Value	Description
0b	*Dest_IP_Address* is an IPv6 IP Address.
1b	*Dest_IP_Address* is an IPv4 IP Address.

DoLoopback (1 bit)

Value	Description
0b	Multicast and unicast packets generated using this Address Handle should not be internally switched within same VSI. Internal switching rules between VSIs are based on the switch configuration.
1b	All unicast UDA packets generated using this Address Handle should be internally switched within VSI. All multicast UDA packets should be both sent to the external port, and internally switched within same VSI.

AH_ID (17 bits)

Index of Address Handle as allocated by software. This index together with HMC configuration for the PCI function identifies location of the hardware Address Handle structure in the host memory.

Traffic_Class_or_TOS (8 bits)

This field specifies the IPv4 Type of Service bits (RFC 2474). If these bits represent the IPv4 TOS bits, only the lower 4-bits are valid. These bits are set by software when the address handle is created and transmitted in the IP header of all sent datagrams for this address handle.

Hop_Limit_or_TTL (8 bits)

This field specifies the IPv4 Time-To-Live (TTL) parameter in the IP header (RFC 791). It is initialized by the software.

Dest_IP_Address_0 (32 bits)

This field specifies bits 127 through 96 of the IPv6 IP Address. Reserved when *IPv4_Valid* is set (1b). The most significant byte of this field (Bits 31:24) is the first byte on the wire for this field.

Dest_IP_Address_1 (32 bits)

This field specifies bits 95 through 64 of the IPv6 IP Address. Reserved when *IPv4_Valid* is set (1b).

Dest_IP_Address_2 (32 bits)

This field specifies bits 63 through 32 of the IPv6 IP Address. Reserved when *IPv4_Valid* is set (1b).

Dest_IP_Address_3 (32 bits)

This field specifies the 32-bit IPv4 IP Address (see RFC 791) or the least significant 32-bits of the IPv6 IP Address. The least significant byte of this field (Bits 7:0) is the last byte on the wire for this field.

Src_IP_Address_0 (32 bits)

This field specifies bits 127 through 96 of the IPv6 IP Address. Reserved when *IPv4_Valid* is set (1b). The most significant byte of this field (Bits 31:24) is the first byte on the wire for this field.

Src_IP_Address_1 (32 bits)

This field specifies bits 95 through 64 of the IPv6 IP Address. Reserved when *IPv4_Valid* is set (1b).

Src_IP_Address_2 (32 bits)

This field specifies bits 63 through 32 of the IPv6 IP Address. Reserved when *IPv4_Valid* is set (1b).

Src_IP_Address_3 (32 bits)

This field specifies the 32-bit IPv4 IP Address (see RFC 791) or the least significant 32-bits of the IPv6 IP Address. The least significant byte of this field (Bits 7:0) is the last byte on the wire for this field.

Src_MAC_Address[5-0] (8 bits each)

Specifies the MAC Address associated with the source IP Address. Index 0 is the LSB and byte 5 is the MSB.

ARP_Index (16 bits)

Index into the ARP cache to specify the Ethernet MAC Address to use for this connection. Initialized by software during connection establishment. Allows access to the entry ARP table. There are no reserved index values. This field is valid for unicast destination MAC Addresses only. Multicast MAC Address is calculated by hardware based on the destination IP Address.

Flow_Label (20 bits)

Flow Label field of IPv6 header.

PD_Index (16 bits) and pd_index_high (2 bits)

Protection Domain for this address handle. This specifies which one of the 256K protection domains this address handle belongs too. There are no reserved index values.

Insert_VLAN_Tag (1 bit)

This bit is set to enable VLAN processing on a connection. The tag configured in the *VLAN_Tag* field is used for all processing. If this bit is clear (0b), there is no VLAN insertion or removal performed by the Protocol Engine. Additional VLAN and priority setting can be configured through the VSI associated with the QP. The most significant portion of the VLAN tag carries the user-specified priority.

VLAN_Tag (16 bits)

Specifies one of the 4096 VLAN tags for this connection, Three bits of priority and one bit canonical format. All bits are valid. The VLAN is in the lower 12 bits of the VLAN Tag field. The upper three bits are priority.

11.5.3.14 Update PE SDs Descriptor Format

This WQE is used to program HMC segment descriptors (SDs) associated with PE-enabled PCI functions. CQP ensures that PFs only access HMC functions that belong to the PF at the time of issues the CQP request. PFs can access its own SDs or SDs of RDMA-enabled VFs that belong to the PF. Accesses to other HMC functions are denied and generate a completion error. This operation is restricted to PF CQP operations.

Table 11-41. CQP Update PE SDs WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:32]	RSVD	[31:0]	SDCMD0
8	[63:32]	SDDATAHIGH0	[31:0]	SDDATALOW0
16	[63:7] [6]	SD_Buffer_Address RSVD	[5:0]	HMC_FCN_ID (PF only)
24	[63] [62:42] [41:38] [37:32]	WQE_Valid RSVD RSVD (AdditionalFragmentCount) OP	[31:8] [7] [6:4] [3:0]	RSVD SKIP_SD_ENTRY_0 RSVD SD_ENTRY_COUNT
32	[63] [62:32]	SD_ENTRY_VALID1 RSVD	[31:0]	SDCMD1
40	[63:32]	SDDATAHIGH1	[31:0]	SDDATALOW1
48	[63] [62:32]	SD_ENTRY_VALID2 RSVD	[31:0]	SDCMD2
56	[63:32]	SDDATAHIGH2	[31:0]	SDDATALOW2

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

HMC_FCN_ID (6 bits)

This field specifies the HMC function ID for the FPM settings to be queried. For PFs, the *HMC_FCN_ID* is strictly the PF index. For VFs, the HMC function ID was returned from the manage HMC PM function CQP operation.

SD_CMD*n* (32 bits)

Table 11-42. SD_CMD Format

Byte Offset	[Bit Range]	Field Name	Description
0	[32:12]	RSVD	Reserved.
	[11:0]	PMSDIDX	Index of the HMC Segment Descriptor.

SD_DATA_LOW*n* (32 bits)

Table 11-43. SD_DATA_LOW Format

Byte Offset	[Bit Range]	Field Name	Description
0	[32:12]	PMSDDATALOW	Bits 31-12 of the segment descriptor.
	[11:2]	PMSDBPCOUNT	When PMSDTYPE is 0, this field has the number of Page Descriptors in this SD.
	[1]	PMSDTYPE	0b = SD points to a page of 512 Page Descriptors. 1b = SD has the physical address of a physically contiguous region.
	[0]	PMSDVALID	Marks the segment descriptor as valid or invalid.

SD_DATA_HIGH*n* (32 bits)

Table 11-44. SD_DATA_HIGH Format

Byte Offset	[Bit Range]	Field Name	Description
0	[32:0]	PMSDDATAHIGH	Most significant 32 bits of a segment descriptor.

SD_ENTRY_VALID*n* (1 bit)

Indicates the SDCMD and DATA information is used to update additional SD entries.

SKIP_SD_ENTRY_0 (1 bit)

Indicates *SD_CMD0*, *SD_DATA_LOW*/*HIGH0* are to be ignored. This is typically useful only if all the SD updates are specified in the update SD host memory structure instead of this WQE.

SD_Buffer_Address (57 bits)

Most significant address bits of the physical address of the update SD host memory structure (listed in Table 11-45). This structure is used to provide additional SD updates. This host memory is only accessed if *SD_ENTRY_COUNT* is not equal to 0b. The number of entries used from the update SD host memory structure is indicated by *SD_ENTRY_COUNT*.

SD_ENTRY_COUNT (4 bits)

Indicates the number of contiguous (starting with *SDCMD3*) SD updates found in the update SD host memory structure.

Table 11-45. Update SD Host Memory Structure

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:32]	RSVD	[31:0]	SDCMD3
8	[63:32]	SDDATAHIGH3	[31:0]	SDDATALOW3
16	[63:32]	RSVD	[31:0]	SDCMD4
24	[63:32]	SDDATAHIGH4	[31:0]	SDDATALOW4
32	[63:32]	RSVD	[31:0]	SDCMD5
40	[63:32]	SDDATAHIGH5	[31:0]	SDDATALOW5
48	[63:32]	RSVD	[31:0]	SDCMD6
56	[63:32]	SDDATAHIGH6	[31:0]	SDDATALOW6
64	[63:32]	RSVD	[31:0]	SDCMD7

Table 11-45. Update SD Host Memory Structure [continued]

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
72	[63:32]	SDDATAHIGH7	[31:0]	SDDATALOW7
80	[63:32]	RSVD	[31:0]	SDCMD8
88	[63:32]	SDDATAHIGH8	[31:0]	SDDATALOW8
96	[63:32]	RSVD	[31:0]	SDCMD9
104	[63:32]	SDDATAHIGH9	[31:0]	SDDATALOW9
112	[63:32]	RSVD	[31:0]	SDCMD10
120	[63:32]	SDDATAHIGH10	[31:0]	SDDATALOW10

11.5.3.15 Query FPM Values Descriptor Format

This WQE is used to query the FPM configuration for the PE portion of the HMC objects. The query FPM values command is used to trigger firmware to return the FPM base registers based on the previous software settings or default values.

Table 11-46. CQP Query FPM Values WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	RSVD		
8	[63:0]	RSVD		
16	[63:6]	RSVD	[5:0]	HMC_FCN_ID (PF only)
24	[63] [62:42] [41:38]	WQE_Valid RSVD RSVD (AdditionalFragmentCount)	[37:32] [31:0]	OP RSVD
32	[63:2]	Physical_Buffer_Address	[1:0]	RSVD
40	[63:0]	RSVD		
48	[63:0]	RSVD		
56	[63:0]	RSVD		

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

HMC_FCN_ID (6 bits)

This field specifies the HMC function ID for the FPM settings to be queried. For PFs, the *HMC_FCN_ID* is strictly the PF index. For VFs, the HMC function ID was returned from the manage HMC PM function CQP operation. If a CQP instance associated with a VF submits this operation, this field is ignored and the *HMC_FCN_ID* is determined by firmware.

Physical_Buffer_Address (62 bits)

This field specifies the physical address of the buffer that firmware fills with the object maximum counts and the object size values.

Table 11-47. FPM Query Configuration Structure Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:45] [44:32]	RSVD max_pe_sds	[31:14] [13:0]	RSVD first_pe_sd_index
8	[63:32] [31:19]	GLHMC_PEQPOBJSZ RSVD	[18:0]	max_qps
16	[63:32] [31:20]	GLHMC_PECQOBJSZ RSVD	[19:0]	max_cqs
24	[63:0]	RSVD		
32	[63:32]	GLHMC_PEHTEOBJSZ	[31:0]	GLHMC_PEHTMAX
40	[63:32]	GLHMC_PEARPOBJSZ	[31:0]	GLHMC_PEARPMAX
48	[63:32]	GLHMC_PEMROBJSZ	[31:0]	GLHMC_PEMRMAX
56	[63:32]	GLHMC_PEXFOBJSZ	[31:0]	GLHMC_PEXFMAX
64	[63:32]	XFBLOCKSIZE	[31:0]	GLHMC_PEXFFLMAX
72	[63:32]	GLHMC_PEQ1OBJSZ	[31:0]	GLHMC_PEQ1MAX
80	[63:32]	Q1BLOCKSIZE	[31:0]	GLHMC_PEQ1FLMAX
88	[63:32]	GLHMC_PETIMEROBJSZ	[31:0]	GLHMC_PETIMERMAX
96	[63:32]	GLHMC_FSIMCOBJSZ	[31:0]	GLHMC_FSIMCMAX
104	[63:32]	GLHMC_FSIAVOBJSZ	[31:0]	GLHMC_FSIAVMAX
112	[63:32]	RSVD	[31:0]	GLHMC_PEPBLMAX
120	[63:48] [47:32] [31:20]	RSVD TIMERBUCKETCNT RSVD	[19:16] [15:10] [9:0]	HTMULTIPLIER RSVD max_ceqs
128	[63:32]	GLHMC_PERRFOBJSZ	[31:0]	GLHMC_PERRFMAX
136	[63:32]	RRFBLOCKSIZE	[31:0]	GLHMC_PERRFFLMAX
144	[63:32]	GLHMC_PEHDROBJSZ	[31:0]	GLHMC_PEHDRMAX
152	[63:32]	GLHMC_PEMDOBJSZ	[31:0]	GLHMC_PEMDMAX
160	[63:32]	GLHMC_PEOOISCOBJSZ	[31:0]	GLHMC_PEOOISCMAX
168	[63:32]	OOISCFBLOCKSIZE	[31:0]	GLHMC_PEOOISCFFLMAX

first_pe_sd_index (14 bits)

This field specifies the function relative starting HMC segment descriptor index for PE resources.

max_pe_sds (13 bits)

This field specifies the maximum number of PE SDs allowed for the HMC function.

max_qps (19 bits)

This field specifies the maximum number of PE QPs allowed for the HMC function.

max_cqs (20 bits)

This field specifies the maximum number of PE CQs allowed for the HMC function.

max_ceqs (10 bits)

This field specifies the maximum number of PE CEQs allowed for the HMC function.

XFBLOCKSIZE (32 bits)

This field specifies the number of transmit FIFO entries per transmit FIFO free list entry. This field is used to determine how much space is consumed for transmit FIFO free list entries. The number of transmit FIFO free list entries is calculated with the following equation:

number of transmit FIFO entries / *XFBLOCKSIZE*

Q1BLOCKSIZE (32 bits)

This field specifies the number of Q1 FIFO entries per Q1 FIFO free list entry. This field is used to determine how much space is consumed for Q1 free list entries. The number of Q1 free list entries is calculated with the following equation:

number of Q1 entries / *Q1BLOCKSIZE*

HTMULTIPLIER (4 bits)

This field specifies the number of hash filter bucket entries per QP. This field is used to determine how much space is consumed for hash filter entries. The number of hash entries is calculated with the following equation:

(round_up_512 (number of QPs + number of Multicast Groups) rounded up to the next power of two) * *HTMULTIPLIER*

TIMERBUCKETCNT (16 bits)

This field specifies the number of timer buckets. This field is used to determine how much space is consumed for timers. The number of timer entries is calculated with the following equation:

(round_up_512 (number of QPs) / 512 + 1) * *TIMERBUCKETCNT*

The remaining fields are register values from the HMC function specified by *HMC_FCN_ID* or the VF index.

RRFBLOCKSIZE (32 bits)

This field specifies the number of Read Response FIFO entries per Read Response FIFO free list entry. This field is used to determine how much space is consumed for Read Response FIFO free list entries. The number of Read Response FIFO free list entries is calculated with the following equation:

number of Read Response FIFO entries / *RRFBLOCKSIZE*

OOISCFBLOCKSIZE (32 bits)

This field specifies the number of Out of Order Send Completion FIFO entries per Out of Order Send Completion FIFO free list entry. This field is used to determine how much space is consumed for Out of Order Send Completion FIFO free list entries. The number of Out of Order Send Completion FIFO free list entries is calculated with the following equation:

number of Out of Order Send Completion FIFO entries / *OOISCFBLOCKSIZE*

11.5.3.16 Commit FPM Values Descriptor Format

This WQE is used to commit the FPM configuration for the PE portion of the HMC objects. The Commit FPM Values command is used to trigger firmware to calculate the FPM base registers based on the software settings.

The Commit FPM Values operation can be applied only once per function. Subsequent uses of this operation for the same function fail with an Invalid State CQP return code. However, the existing values for the function are returned as output in the buffer.

Table 11-48. CQP Commit FPM Values WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	RSVD		
8	[63:0]	RSVD		
16	[63:6]	RSVD	[5:0]	HMC_FCN_ID (PF only)
24	[63] [62:42] [41:38]	WQE_Valid RSVD RSVD (AdditionalFragmentCount)	[37:32] [31:0]	OP RSVD
32	[63:2]	Physical_Buffer_Address	[1:0]	RSVD
40	[63:0]	RSVD		
48	[63:0]	RSVD		
56	[63:0]	RSVD		

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

HMC_FCN_ID (6 bits)

This field specifies the HMC function ID for the FPM settings to be programmed. For PFs, the *HMC_FCN_ID* is strictly the PF index. For VFs, the HMC function ID was returned from the manage HMC PM function CQP operation. If a CQP instance associated with a VF submits this operation, this field is ignored and the *HMC_FCN_ID* is determined by firmware.

Physical_Buffer_Address (62 bits)

This field specifies the most significant bits of the physical address of the buffer that firmware uses to configure the object maximum counts and the object size values. The buffer described must be aligned to a 4-byte aligned boundary.

Table 11-49. FPM Commit Configuration Structure Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:32] [31:19]	GLHMC_PEQPBASE RSVD	[18:0]	GLHMC_PEQPCNT
8	[63:32] [31:20]	GLHMC_PECQBASE RSVD	[19:0]	GLHMC_PECQCNT
16	[63:0]	RSVD		
24	[63:32]	GLHMC_PEHTEBASE	[31:0]	GLHMC_PEHTCNT[1]
32	[63:32]	GLHMC_PEARPBASE	[31:0]	GLHMC_PEARPCNT
40	[63:32]	GLHMC_APBVTINUSEBASE	[31:0]	RSVD

Table 11-49. FPM Commit Configuration Structure Format [continued]

Byte Offset	[Bit Range]	Field Name		
48	[63:32]	GLHMC_PEMRBASE	[31:0]	GLHMC_PEMRCNT
56	[63:32]	GLHMC_PEXFBASE	[31:0]	GLHMC_PEXFCNT
64	[63:32]	GLHMC_PEXFFLBASE	[31:0]	GLHMC_PEXFFLCNT[1]
72	[63:32]	GLHMC_PEQ1BASE	[31:0]	GLHMC_PEQ1CNT
80	[63:32]	GLHMC_PEQ1FLBASE	[31:0]	GLHMC_PEQ1FLCNT[1]
88	[63:32]	GLHMC_PETIMERBASE	[31:0]	GLHMC_PETIMERCNT[1]
96	[63:32]	GLHMC_FSIMCBASE	[31:0]	GLHMC_FSIMCCNT
104	[63:32]	GLHMC_FSIAVBASE	[31:0]	GLHMC_FSIAVCNT
112	[63:32]	GLHMC_PEPBLBASE	[31:0]	GLHMC_PEPBLCNT
120	[63:0]	RSVD		
128	[63:32]	GLHMC_PERRFBASE	[31:0]	GLHMC_PERRFCNT
136	[63:32]	GLHMC_PERRFFLBASE	[31:0]	GLHMC_PERRFFLCNT[1]
144	[63:32]	GLHMC_PEHDRBASE	[31:0]	GLHMC_PEHDRCNT
152	[63:32]	GLHMC_PEMDBASE	[31:0]	GLHMC_PEMDCNT
160	[63:32]	GLHMC_PEOOISCBASE	[31:0]	GLHMC_PEOOISCCNT
168	[63:32]	GLHMC_PEOOISCFFLBASE	[31:0]	GLHMC_PEOOISCFFLCNT[1]

1. These fields are calculated by CQP and returned on completion. On submission, these fields are ignored.

These fields are register values from the HMC function specified by *HMC_FCN_ID* or the *VF_Index*. On request submission, the GLHMC_{obj}CNT fields must be filled in the software. Upon completion of the request, all fields are filled in with the updated values that were actually committed.

11.5.3.17 Flush WQEs Descriptor Format

This WQE is used to flush pending WQEs to a CQ. The initial usage for the WQE is to support Winsock Direct socket hand-off from one process to another. This WQE is now also used for all WQE flushing. Host software uses this after the quad hash has been deleted for a QP in order to comply with verbs requirements. Note that only a single unprocessed WQE is flushed from the WQ(s) specified by the *FlushRQ*/*FlushSQ* bits. Host software must generate additional completions if more than one outstanding WQE is pending on a WQ that has been flushed.

This WQE can be used to flush pending transmit and receive WQEs for UDA and RDMA QPs. For UDA all filters forwarding to the specified UDA QP must be disabled, prior to flushing WQEs.

Table 11-50. CQP Flush WQEs WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	RSVD		
8	[63:20] [19:16]	RSVD AE_Source	[15:12] [11:0]	RSVD AE_Code
16	[63:48] [47:32]	SQ_Major_Code SQ_Minor_Code	[31:16] [15:0]	RQ_Major_Code RQ_Minor_Code
24	[63] [62] [61] [60] [59]	WQE_Valid FlushRQ FlushSQ UserFlushCode GenerateAE	[58:42] [41:38] [37:32] [31:18] [17:0]	RSVD RSVD (AdditionalFragmentCount) OP RSVD QP_ID
32	[63:0]	RSVD		
40	[63:0]	RSVD		
48	[63:0]	RSVD		
56	[63:0]	RSVD		

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

FlushRQ (1 bit)

Value	Description
0b	Do not flush any pending WQEs from the RQ.
1b	Generate a completion for the first pending RQ with a flushed return code.

FlushSQ (1 bit)

Value	Description
0b	Do not flush any pending WQEs from the SQ.
1b	Generate a completion for the first pending SQ with a flushed return code.

UserFlushCode (1 bit)

UserFlushCode indicates that the WQEs flushed should carry the values specified by SQ/RQ major and minor codes.

Value	Description
0b	Any flushed WQEs carry the flushed completion major and minor codes.
1b	Flushed WQEs for the SQ carry *SQ_Major_Code* and *SQ_Minor_Code*. Flushed WQEs on the RQ carry *RQ_Major_Code* and *RQ_Minor_Code*.

SQ_Major_Code (16 bits)

Valid only when *UserFlushCode*=1b. *SQ_Major_Code* is reported in *Major_Error_Code* field of the CQE for the first flushed WQE for the SQ of the QP specified by *QP_ID*.

SQ_Minor_Code (16 bits)

Valid only when *UserFlushCode*=1b. *SQ_Minor_Code* is reported in *Minor_Error_Code* field of the CQE for the first flushed WQE for the SQ of the QP specified by *QP_ID*.

RQ_Major_Code (16 bits)

Valid only when *UserFlushCode*=1b. *RQ_Major_Code* is reported in *Major_Error_Code* field of the CQE for the first flushed WQE for the RQ of the QP specified by *QP_ID*.

RQ_Minor_Code (16 bits)

Valid only when *UserFlushCode*=1. *RQ_Minor_Code* is reported in *Minor_Error_Code* field of the CQE for the first flushed WQE for the RQ of the QP specified by *QP_ID*.

GenerateAE (1 bit)

GenerateAE indicates that an AE should be generated after the CQEs have been generated for the flushed WQEs.

AE_Code (12 bits)

Valid only when *GenerateAE*=1. See Table 11-12 for the values that should be used for this field.

AE_Source (4 bits)

Valid only when *GenerateAE*=1. See Section 11.4.6 for the values that should be used for this field.

QP_ID (18 bits)

Identifies the QP number targeted for the flush operation. This must be an accelerated QP (UDA or iWARP or RoCEv2).

11.5.3.18 Manage Accelerated Port Bit Vector (APBV)

This WQE sets or clears the bit in the APBV table for the host NIC, IP Address, and TCP or UDP port number specified in the WQE. The APBV table is used to filter TCP segments belonging to accelerated RDMA connections, TCP segments, UDP multicast and unicast datagrams belonging to accelerated UDA QPs. Host software also uses the APBV table to flag inbound TCP packets used for connection setup of accelerated connections.

Table 11-51. CQP Manage Accelerated Port Table WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	RSVD		
8	[63:0]	RSVD		
16	[63:16]	RSVD	[15:0]	Local_TCP/UDP_Port
24	[63] [62] [61:42]	WQE_Valid Add_Port RSVD	[41:38] [37:32] [31:0]	RSVD (AdditionalFragmentCount) OP RSVD
32	[63:0]	RSVD		
40	[63:0]	RSVD		
48	[63:0]	RSVD		
56	[63:0]	RSVD		

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

Add_Port (1 bit)

Value	Description
0b	The port is deleted from the table.
1b	The port is added to the table.

Local_TCP/UDP_Port (16 bits)

The local (destination port for inbound packets) TCP/IP or UDP/IP port that is used for accelerated RDMA connections and accelerated UDA UDP unicast and multicast datagrams, and UDA TCP streams. Any inbound TCP packet and UDP datagram who's destination IP Address matches that IP Address table and destination TCP/UDP port matches *Local_TCP/UDP_Port* is filtered through quad hash table. If missed, TCP packet and UDP datagram is passed to the host NIC.

11.5.3.19 NOP Descriptor Format

This WQE can be used to generate a CQE that can be used to trigger subsequent processing in a completion handler or as a synchronization mechanism to indicate when all previously issued CQP requests have completed.

Table 11-52. CQP NOP WQE Format

Byte Offset	[Bit Range] Field Name			
0	[63:0]	RSVD		
8	[63:0]	RSVD		
16	[63:0]	RSVD		
24	[63] [62:42] [41:38]	WQE_Valid RSVD RSVD (AdditionalFragmentCount)	[37:32] [31:0]	OP RSVD
32	[63:0]	RSVD		
40	[63:0]	RSVD		
48	[63:0]	RSVD		
56	[63:0]	RSVD		

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

11.5.3.20 Manage Quad Hash Table Descriptor Format

The quad hash table entry can be configured to forward packets to the UDA QP. Multiple entries of the quad hash table can refer to the same QP or multicast group.

The same quad hash table is intended to be shared by RDMA QPs and UDA QPs and multicast groups. The quad hash table must be properly sized to allow a good hit rate. It also must be properly sized with respect to the maximum number of entries that are intended to be allocated. One approach is to limit the total number of resources that can be allowed to use quad hash for the given function. This would enable the best use of the shared table. In any case, since allocation of entries in the quad hash table is controlled by PCI function driver, sizing of the table, and allocation policy should be controlled there.

UDA UDP and TCP traffic should be using quad hash tables associated with respective PCI function (PF or VF).

The quad hash table index is calculated by hardware. All hash functions include destination MAC Address and VLAN (if valid) in addition to the description provided later. Hash function inputs and a kind of hash function depends on a hash table entry type:

- **UDP Unicast** — For the first UDP unicast fragment, or UDP unicast datagram, Hash Function should use as an input a destination IP Address and destination UDP port. Hash Table entry should match both destination IP Address and destination UDP port, and should be of UDP Unicast entry type. Matching entry should carry a destination Queue Pair Number.
- **TCP SYN (ACK Clear)** — For the TCP packets with the SYN bit set, a hash function should be used as an input a destination IP Address and destination TCP port. A hash table entry should match both destination IP Address and destination TCP port, and should be of TCP SYN (ACK clear) entry type. Matching entry should carry a destination QP number.

- **Established TCP** — For the TCP packets with the SYN bit clear or the *SYN* bit set and *ACK* bit set, the hash function should be used as an input source and destination IP Addresses, and source and destination TCP ports. The hash table should match both source and destination IP Addresses, and source and destination TCP ports. Matching entry should carry a destination QP number.
- **RoCEv2 Multicast** — RoCEv2 multicast packets go through the Quad Hash to look up the multicast group number. Hash Table entry should match the destination IP Address, the destination QP has all bits on (1's), and should be of multicast entry type. Matching entry should carry a multicast group number. RoCEv2 unicast packets do not go through the Quad Hash.

Table 11-53. CQP Manage Quad Hash Table WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:48] [47:40] [39:32] [31:24]	RSVD Dest_MAC_Address[5] Dest_MAC_Address[4] Dest_MAC_Address[3]	[23:16] [15:8] [7:0]	Dest_MAC_Address[2] Dest_MAC_Address[1] Dest_MAC_Address[0]
8	[63:50] [49:32]	RSVD QPN/MGN	[31:16] [15:0]	src_port dest_port
16	[63:44] [43:32]	RSVD VLAN_ID	[31:10] [9:0]	RSVD QS_Handle
24	[63] [62:61] [60] [59] [58:45]	WQE_Valid ManageEntry IPv4_Valid VLAN_Valid RSVD	[44:42] [41:38] [37:32] [31:0]	EntryType RSVD (AdditionalFragmentCount) OP RSVD
32	[63:32]	Src_IP_Address_2	[31:0]	Src_IP_Address_3
40	[63:32]	Src_IP_Address_0	[31:0]	Src_IP_Address_1
48	[63:32]	Dest_IP_Address_2	[31:0]	Dest_IP_Address_3
56	[63:32]	Dest_IP_Address_0	[31:0]	Dest_IP_Address_1

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

QS_Handle (10 bits)

Identifies the queue set handle associated with the hash entry. See Section 8.3.3.4 for more information on scheduler configuration.

Dest_IP_Address_0 (32 bits)

This field specifies bits 127 through 96 of the IPv6 IP Addresses. Reserved when *IPv4_Valid* is set (1b). The MSB of this field (Bits 31:24) is the first byte on the wire for this field.

Dest_IP_Address_1 (32 bits)

This field specifies bits 95 through 64 of the IPv6 IP Addresses. Reserved when *IPv4_Valid* is set (1b).

Dest_IP_Address 2 (32 bits)

This field specifies bits 63 through 32 of the IPv6 IP Addresses. Reserved when *IPv4_Valid* is set (1b).

Dest_IP_Address_3 (32 bits)

This field specifies the 32-bit IPv4 (see RFC 791) or the least significant 32-bits of the IPv6 IP Address. The LSB of this field (Bits 7:0) is the last byte on the wire for this field.

Src_IP_Address_0 (32 bits)

This field specifies bits 127 through 96 of the IPv6 IP Addresses. Reserved when *IPv4_Valid* is set (1b). The MSB of this field (Bits 31:24) is the first byte on the wire for this field.

Src_IP_Address_1 (32 bits)

This field specifies bits 95 through 64 of the IPv6 IP Addresses. Reserved when *IPv4_Valid* is set (1b).

Src_IP_Address_2 (32 bits)

This field specifies bits 63 through 32 of the IPv6 IP Addresses. Reserved when *IPv4_Valid* is set (1b).

Src_IP_Address_3 (32 bits)

This field specifies the 32-bit IPv4 (see RFC 791) or the least significant 32-bits of the IPv6 IP Address. The LSB of this field (Bits 7:0) is the last byte on the wire for this field.

src_port (16 bits)

Source UDP/TCP port.

dest_port (16 bits)

Destination UDP/TCP port.

QPN/MGN (18 bits)

Index of the QP context or Multicast Group Context associated with the entry. When modifying a quad hash entry for a Multicast Group, the Multicast Group Number is smaller than the QP number. The high bits must be filled with leading zeros.

EntryType (3 bits)

The quad hash table is used as a perfect filter for multiple types of accelerated traffic (Userspace TCP, RoCEv2, Userspace UDP unicast and multicast). Depending on the traffic type different hardware accelerations might apply.

Value	Description
001b	Userspace TCP Established
010b	Userspace TCP SYN (ACK clear)
011b	Userspace UDP Unicast
110b	Userspace UDP Multicast
Note: All other values are reserved	

IPv4_Valid (1 bit)

Value	Description
0b	Indicates that both source and destination IP Addresses are IPv6.
1b	Indicates that both source and destination IP Addresses are IPv4.

ManageEntry (2 bits)

Value	Description
00b	Delete entry from the table.
01b	Add entry to the table.
10b	Modify entry.
11b	Reserved.

Dest_MAC_Address[5-0] (8 bits each)

Specifies the MAC Address associated with the destination IP Address. Index 0 is the LSB and byte 5 is the MSB.

VLAN_ID (12 bits)

Specifies the VLAN ID associated with the hash entry. This field is ignored unless *VLAN_Valid* is set (1b).

VLAN_Valid (1 bit)

Specifies if the hash entry has a VLAN associated with it.

Value	Description
0b	The hash entry is not associated with a VLAN.
1b	The VLAN associated with the hash entry is specified in the *VLAN_ID* field.

11.5.3.21 Create/Modify/Destroy Multicast Group Descriptor Format

This WQE can be used to Create/Modify/Destroy Multicast Group. This WQE can be used only by Physical PCI Function driver. Software owns content of the Multicast Group Context. Every time new QP subscribes or leaves the multicast group, driver should Modify the contents of the Multicast Group Context and issue the Modify Multicast Group descriptor to CQP.

The Create Multicast Group creates an entry in the Quad Hash. The Destroy Multicast Group removes the Quad Hash entry. Modify Multicast Group does not modify the Quad Hash entry.

The Modify Multicast Group requires *HMC_FCN_ID*, *MG_ID* and an updated MG context. The other fields are ignored.

Table 11-54. CQP Create/Modify/Destroy Multicast Group WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:48] [47:40] [39:32] [31:24]	RSVD Dest_MAC_Address[5] Dest_MAC_Address[4] Dest_MAC_Address[3]	[23:16] [15:8] [7:0]	Dest_MAC_Address[2] Dest_MAC_Address[1] Dest_MAC_Address[0]
8	[63:6]	RSVD	[5:0]	HMC_FCN_ID
16	[63:44] [43:32]	RSVD VLAN_ID	[31:16] [9:0]	RSVD QS_Handle
24	[63] [62:61] [60] [59]	WQE_Valid RSVD IPv4_Valid VLAN_Valid	[58:38] [37:32] [31:13] [12:0]	RSVD (AdditionalFragmentCount) OP RSVD MG_ID
32	[63:0]	MGContextAddress		
40	[63:0]	RSVD		
48	[63:32]	Dest_IP_Address_2	[31:0]	Dest_IP_Address_3
56	[63:32]	Dest_IP_Address_0	[31:0]	Dest_IP_Address_1

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

MG_ID (13 bits)

Index of the Multicast Group. Allocated and managed by physical function driver. Each PCI function has a dedicated zero-based Multicast Group IDs space.

MGContextAddress (64 bits)

Physical address of the software Multicast Group context. *MGContextAddress* should be aligned to 64 bytes.

HMC_FCN_ID (6 bits)

This field specifies the HMC function ID for the FPM settings to be queried. For PFs, the *HMC_FCN_ID* is strictly the PF index. For VFs, the HMC function ID was returned from the manage HMC PM function CQP operation.

IPv4_Valid (1 bits)

Value	Description
0b	Indicates that both source and destination IP Addresses are IPv6.
1b	Indicates that both source and destination IP Addresses are IPv4.

Dest_IP_Address_0 (32 bits)

This field specifies bits 127 through 96 of the IPv6 IP Addresses. Reserved when *IPv4_Valid* is set (1b). The MSB of this field (Bits 31:24) is the first byte on the wire for this field.

Dest_IP_Address_1 (32 bits)

This field specifies bits 95 through 64 of the IPv6 IP Addresses. Reserved when *IPv4_Valid* is set (1b).

Dest_IP_Address 2 (32 bits)

This field specifies bits 63 through 32 of the IPv6 IP Addresses. Reserved when *IPv4_Valid* is set (1b).

Dest_IP_Address_3 (32 bits)

This field specifies the 32-bit IPv4 (see RFC 791) or the least significant 32-bits of the IPv6 IP Address. The LSB of this field (Bits 7:0) is the last byte on the wire for this field.

Dest_MAC_Address[5-0] (8 bits each)

Specifies the MAC Address associated with the destination IP Address. Index 0 is the LSB and byte 5 is the MSB.

VLAN_ID (12 bits)

Specifies the VLAN ID associated with the hash entry. This field is ignored unless *VLAN_Valid* is set (1b).

VLAN_Valid (1 bit)

Specifies if the hash entry has a VLAN associated with it.

Value	Description
0b	The hash entry is not associated with a VLAN.
1b	The VLAN associated with the hash entry is specified in the *VLAN_ID* field.

QS_Handle (10 bits)

This field specifies Tx-Scheduler QS handle associated with the TC for this QP. See Section 8.3.3.4 for more information on scheduler configuration. For VFs, the *QS_Handle* is checked to ensure that the VF issuing the CQP command is associated with the *QS_Handle*. *QS_Handle* is valid only for Create Multicast Group.

11.5.3.22 Suspend QP Descriptor Format

This WQE can be used to suspend a QP that is to be moved to a different QS than the one to which it is currently assigned. QP suspension is required before changing the QS. This WQE might return suspend pending in which case an AE_QP_SUSPEND_COMPLETE AE is issued when the suspend operation completed.

Table 11-55. CQP Suspend QP WQE Format

Byte Offset	[Bit Range] Field Name			
0	[63:0] RSVD			
8	[63:0] RSVD			
16	[63:0] RSVD			
24	[63] WQE_Valid [62:42] RSVD [41:38] RSVD (AdditionalFragmentCount)		[37:32] OP [31:18] RSVD [17:0] QP_ID	
32	[63:0] RSVD			
40	[63:0] RSVD			
48	[63:0] RSVD			
56	[63:0] RSVD			

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

QP_ID (18 bits)

Index of the QP to be suspended.

11.5.3.23 Resume QP Descriptor Format

This WQE can be used to resume a QP that had been previous suspended in order to be moved to a different QS than the one to which it was assigned. QP suspension is required before changing the QS.

Table 11-56. CQP Resume QP WQE Format

Byte Offset	[Bit Range]	Field Name		
8	[63:0]	RSVD		
8	[63:0]	RSVD		
16	[63:10]	RSVD	[9:0]	QS_Handle
24	[63] [62:42] [41:38]	WQE_Valid RSVD RSVD (AdditionalFragmentCount)	[37:32] [31:18] [17:0]	OP RSVD QP_ID
32	[63:0]	RSVD		
40	[63:0]	RSVD		
48	[63:0]	RSVD		
56	[63:0]	RSVD		

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

QP_ID (18 bits)

Index of the QP to be suspended.

QS_Handle (10 bits)

This field specifies Tx-Scheduler QS handle associated with the TC for this QP. See Section 8.3.3.4 for more information on scheduler configuration. For VFs, the *QS_Handle* is checked to ensure that the VF issuing the CQP command is associated with the *QS_Handle*.

11.5.3.24 Static HMC Resources Allocated Descriptor Format

This WQE must be used after the commit FPM values operation is complete and all of the static HMC resources from Table 11-14 have HMC backing pages allocated and configured. CQP then completes initializing the function for PE operation.

Table 11-57. Static HMC Resources Allocated WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	RSVD		
8	[63:0]	RSVD		
16	[63:6]	RSVD	[5:0]	HMC_FCN_ID (PF only)
24	[63] [62:42] [41:38]	WQE_Valid RSVD RSVD (AdditionalFragmentCount)	[37:32] [31:0]	OP RSVD
32	[63:0]	RSVD		
40	[63:0]	RSVD		
48	[63:0]	RSVD		
56	[63:0]	RSVD		

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

HMC_FCN_ID (6 bits)

This field specifies the HMC function ID that software has completed the allocation and configuration of the static HMC backing pages. For PFs, the *HMC_FCN_ID* is strictly the PF index. For VFs, the HMC function ID was returned from the manage HMC PM function CQP operation. If a CQP instance associated with a VF submits this operation, this field is ignored and the *HMC_FCN_ID* is determined by firmware.

11.5.3.25 Manage Statistics Instance Descriptor Format

Every function (PF or VF) is assigned a statistics instance. By default, all QPs in that function uses the one assigned to its function. However, software is able to allocate a new statistics instance that can be used for one or a set of QPs within the function. For example, all QPs belonging to a VSI.

This WQE is used to allocate or free a statistics instance. This WQE can be used only by the Physical PCI Function driver.

Table 11-58. Manage Statistics Instance WQE Format

Byte Offset	[Bit Range]	Field Name		
0	[63:0]	RSVD		
8	[63:0]	RSVD		
16	[63:0]	RSVD		
24	[63] [62] [61] [60] [59:42]	WQE_Valid alloc_stats_instance RSVD use_hmc_fcn_index RSVD	[41:38] [37:32] [31:7] [6:0]	RSVD (AdditionalFragmentCount) OP RSVD Statistics_Instance_Index
32	[63:0]	RSVD		
40	[63:6]	RSVD	[5:0]	hmc_fcn_index
48	[63:0]	RSVD		
56	[63:0]	RSVD		

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

alloc_stats_instance (1 bit)

Value	Description
0b	The statistics set is deallocated.
1b	A new statistics set is to be allocated. If successful, the number of the statistics set is returned in *Operation_Return_Value*.

use_hmc_fcn_index (1 bit)

Value	Description
0b	The HMC function index is determined by the PCI function associated with the PCI function that issues the CQP operation.
1b	*hmc_fcn_index* can be set to a HMC function index that is a VF associated with the PF associated with the CQP instance used to issue the command.

Only valid when issued to a PF CQP instance. Ignored (treated as if set to 0b) for VF CQP operations.

hmc_fcn_index (6 bits)

Only valid when issued to a PF CQP instance and *use_hmc_fcn_index* is set to 1b. Ignored otherwise.

Statistics_Instance_Index (7 bits)

Statistics Set to be deleted. Only valid when *alloc_stats_instance* is set to 0b.

11.5.3.26 Gather Statistics Descriptor Format

This WQE is used to gather the statistics for a function (PF or VF). This WQE can be used only by the Physical PCI Function driver.

Table 11-59. Gather Statistics Instance WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	RSVD		
8	[63:0]	RSVD		
16	[63:0]	RSVD		
24	[63] [62] [61] [60]	WQE_Valid RSVD Use_Statistics_Instance use_hmc_fcn_index	[59:38] [37:32] [31:7] [6:0]	RSVD (AdditionalFragmentCount) OP RSVD Statistics_Instance_Index
32	[63:0]	Physical_Buffer_Address		
40	[63:6]	RSVD	[5:0]	hmc_fcn_index
48	[63:0]	RSVD		
56	[63:0]	RSVD		

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

use_hmc_fcn_index (1 bit)

Value	Description
0b	The HMC function index is determined by the PCI function associated with the PCI function that issues the CQP operation.
1b	*hmc_fcn_index* can be set to a HMC function index that is a VF associated with the PF associated with the CQP instance used to issue the command.

Only valid when issued to a PF CQP instance. Ignored (treated as if set to 0) for VF CQP operations.

hmc_fcn_index (6 bits)

Only valid when issued to a PF CQP instance and *use_hmc_fcn_index* is set to 1b. Ignored otherwise.

Physical_Buffer_Address (64 bits)

This field specifies the physical address of the buffer that firmware fills with the statistics.

Use_Statistics_Instance (1 bit)

This field indicates if the default per Private Memory Function statistics are gathered or if one of the additional RDMA statistics instances are gathered for this QP.

Value	Description
0b	The default statistics are used.
1b	The statistics instance indicated by the *Statistics_Instance_Index* field are used.

Statistics_Instance_Index (7 bits)

Statistics Set to be gathered. This field is ignored if *Use_Statistics_Instance* is 0b.

Table 11-60. Statistics Buffer

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:56] [55:32]	RSVD PES_RXVLANERR	[31:0]	RSVD
8	[63:48] [47:32]	RSVD PES_IP4RXOCTSHI	[31:0]	PES_IP4RXOCTSLO
16	[63:48] [47:32]	RSVD PES_IP4RXPKTSHI	[31:0]	PES_IP4RXPKTSLO
24	[63:32]	PES_IP4RXDISCARD	[31:0]	PES_IP4RXTRUNC
32	[63:48] [47:32]	RSVD PES_IP4RXFRAGSHI	[31:0]	PES_IP4RXFRAGSLO
40	[63:48] [47:32]	RSVD PES_IP4RXMCOCTSHI	[31:0]	PES_IP4RXMCOCTSLO
48	[63:48] [47:32]	RSVD PES_IP4RXMCPKTSHI	[31:0]	PES_IP4RXMCPKTSLO
56	[63:48] [47:32]	RSVD PES_IP6RXOCTSHI	[31:0]	PES_IP6RXOCTSLO
64	[63:48] [47:32]	RSVD PES_IP6RXPKTSHI	[31:0]	PES_IP6RXPKTSLO
72	[63:32]	PES_IP6RXDISCARD	[31:0]	PES_IP6RXTRUNC
80	[63:48] [47:32]	RSVD PES_IP6RXFRAGSHI	[31:0]	PES_IP6RXFRAGSLO
88	[63:48] [47:32]	RSVD PES_IP6RXMCOCTSHI	[31:0]	PES_IP6RXMCOCTSLO
96	[63:48] [47:32]	RSVD PES_IP6RXMCPKTSHI	[31:0]	PES_IP6RXMCPKTSLO
104	[63:48] [47:32]	RSVD PES_IP4TXOCTSHI	[31:0]	PES_IP4TXOCTSLO
112	[63:48] [47:32]	RSVD PES_IP4TXPKTSHI	[31:0]	PES_IP4TXPKTSLO
120	[63:48] [47:32]	RSVD PES_IP4TXFRAGSHI	[31:0]	PES_IP4TXFRAGSLO
128	[63:48] [47:32]	RSVD PES_IP4TXMCOCTSHI	[31:0]	PES_IP4TXMCOCTSLO
136	[63:48] [47:32]	RSVD PES_IP4TXMCPKTSHI	[31:0]	PES_IP4TXMCPKTSLO

Table 11-60. Statistics Buffer [continued]

Byte Offset	[Bit Range]	Field Name		
144	[63:48] [47:32]	RSVD PES_IP6TXOCTSHI	[31:0]	PES_IP6TXOCTSLO
152	[63:48] [47:32]	RSVD PES_IP6TXPKTSHI	[31:0]	PES_IP6TXPKTSLO
160	[63:48] [47:32]	RSVD PES_IP6TXFRAGSHI	[31:0]	PES_IP6TXFRAGSLO
168	[63:48] [47:32]	RSVD PES_IP6TXMCOCTSHI	[31:0]	PES_IP6TXMCOCTSLO
176	[63:48] [47:32]	RSVD PES_IP6TXMCPKTSHI	[31:0]	PES_IP6TXMCPKTSLO
184	[63:56] [55:32]	RSVD PES_IP4TXNOROUTE	[31:24] [23:0]	RSVD PES_IP6TXNOROUTE
192	[63:48] [47:32]	RSVD PES_TCPRXSEGSHI	[31:0]	PES_TCPRXSEGSLO
200	[63:56] [55:32]	RSVD PES_TCPRXOPTERR	[31:24] [23:0]	RSVD PES_TCPRXPROTOERR
208	[63:48] [47:32]	RSVD PES_TCPTXSEGHI	[31:0]	PES_TCPTXSEGLO
216	[63:32]	PES_TCPRTXSEG	[31:0]	RSVD
224	[63:48] [47:32]	RSVD PES_UDPRXPKTSHI	[31:0]	PES_UDPRXPKTSLO
232	[63:48] [47:32]	RSVD PES_UDPTXPKTSHI	[31:0]	PES_UDPTXPKTSLO
240	[63:48] [47:32]	RSVD PES_RDMARXWRSHI	[31:0]	PES_RDMARXWRSLO
248	[63:48] [47:32]	RSVD PES_RDMARXRDSHI	[31:0]	PES_RDMARXRDSLO
256	[63:48] [47:32]	RSVD PES_RDMARXSNDSHI	[31:0]	PES_RDMARXSNDSLO
264	[63:48] [47:32]	RSVD PES_RDMATXWRSHI	[31:0]	PES_RDMATXWRSLO
272	[63:48] [47:32]	RSVD PES_RDMATXRDSHI	[31:0]	PES_RDMATXRDSLO
280	[63:48] [47:32]	RSVD PES_RDMATXSNDSHI	[31:0]	PES_RDMATXSNDSLO
288	[63:48] [47:32]	RSVD PES_RDMAVBNDHI	[31:0]	PES_RDMAVBNDLO
296	[63:48] [47:32]	RSVD PES_RDMAVINVHI	[31:0]	PES_RDMAVINVLO
304	[63:48] [47:32]	RSVD PES_RXNPECNMARKEDPKTSHI	[31:0]	PES_RXNPECNMARKEDPKTSLO
312	[63:48] [47:32]	RSVD PES_RXRPCNPIGNORED	[31:0]	PES_RXRPCNPHANDLED
320	[63:32]	RSVD	[31:0]	PES_TXNPCNPSENT
328-1016	[63:0]	RSVD		

11.5.3.27 Manage Work Scheduler (WS) Node Descriptor Format

This WQE can be used to Create/Modify/Destroy a WS Node.

Software must create the root node (*Node_ID*=0) for its port. Intermediate or Leaf Nodes can be attached to the root node by specifying that the *Parent_Node_ID*=0 during the add operation.

For an Add operation of a Leaf node, the QSet handle is returned in the *Operation_Return_Value* if the operation is successful.

For a Modify operation, the only fields that can be modified are *Weight* and *enable_node*.

Table 11-61. CQP Create/Modify/Destroy Work Scheduler Node WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	RSVD		
8	[63:0]	RSVD		
16	[63:0]	RSVD		
24	[63] [62] [61] [60:59] [58:56] [55:52] [51:42]	WQE_Valid enable_node Node_Type Priority_Type Traffic_Class node_op RSVD	[41:38] [37:32] [31:26] [25:16] [15:10] [9:0]	RSVD (AdditionalFragmentCount) OP RSVD Parent_Node_ID RSVD Node_ID
32	[63:58] [57:48] [47] [46:44] [43]	RSVD VSI_Index RSVD failing_port RSVD	[42:40] [39] [38:32] [31:0]	failover_port RSVD Weight RSVD
40	[63:0]	RSVD		
48	[63:0]	RSVD		
56	[63:0]	RSVD		

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

Node_Type (1 bit)

This field indicates if this is a Leaf Node (1b) or a Parent Node (0b). The following fields must be filled in for a Leaf Node: *Priority_Type*, *Traffic_Class*, *VSI_Index*, *Parent_Node_ID*, and *Weight*.

Priority_Type (2 bits)

This field specifies the priority type used for this node:

Value	Description
00b	Reserved
01b	Weighted Round-Robin
10b	Strict Priority
11b	Weighted Strict Priority

Traffic_Class (3 bits)

The port's traffic class represented by this node.

Parent_Node_ID (10 bits)

This field specifies where in the tree to place the new Node. That is, the new Node becomes the child of the node indicated in the *Parent_Node_ID* field. Node IDs are zero relative within a function. Software does not know what port it is on so a value of (0x0) means "this function's root node".

Node_ID (10 bits)

This is the Node ID that will be used to modify and delete the node. This value is assigned by software and must be unique. Software does not know what port it is on so a value of (0x0) means "this functions root node".

node_op (4 bits)

What operation to perform:

Value	Description
0x0	Add
0x1	Modify
0x2	Delete
0x3	failover_start
0x4	failover_complete
Note: All other values are reserved	

Only one failover can be active at a time.

VSI_Index (10 bits)

The VSI index for the WS node. This field is required for all manage work scheduler node operations.

Weight (7 bits)

Weight given to this node expressed as a percentage relative to it siblings. This value is valid only for Weighted Round-Robin and Weighted Strict Priority (zero is not valid).

enable_node (1 bit)

Value	Description
0b	The Work Scheduler node is disabled. This allows a Work Scheduler node to be created but not yet enabled. The node can be enabled/disabled later with the modify Work Scheduler operation.
1b	The Work Scheduler node is enabled.

This is only for leaf nodes. It is ignored for non-leaf nodes.

failing_port (3 bits)

If *node_op*=failover_start, the work scheduler node is being moved from this port. This field is ignored for any other *node_op* value.

failover_port (3 bits)

If *node_op*=failover_start, the work scheduler node is being moved to this port. This field is ignored for any other *node_op* value.

11.5.3.28 Set UP-UP Mapping

This WQE is used on a PF to set the User Priority (UP) mapping table for a VF. Only valid when issued to a PF CQP instance. Ignored otherwise.

Table 11-62. Set UP-UP Mapping WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:56] [55:48] [47:40] [39:32]	UP_mapping[7] UP_mapping[6] UP_mapping[5] UP_mapping[4]	[31:24] [23:16] [15:8] [7:0]	UP_mapping[3] UP_mapping[2] UP_mapping[1] UP_mapping[0]
8	[63:0]	RSVD		
16	[63:0]	RSVD		
24	[63] [62] [61] [60]	WQE_Valid use_VLAN use_CNP_UP_override RSVD	[59:38] [37:32] [31:0]	RSVD (AdditionalFragmentCount) OP RSVD
32	[63:0]	RSVD		
40	[63:38] [37:32]	RSVD CNP_UP_override	[31:6] [5:0]	RSVD hmc_fcn_index
48	[63:0]	RSVD		
56	[63:0]	RSVD		

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

hmc_fcn_index (6 bits)

HMC index of the function for this mapping.

UP_mapping[7-0] (8 bits each)

User priorities indexed by the VF UP number.

use_CNP_UP_override (1 bit)

Value	Description
0b	Indicates that CNP packets are sent using the assigned UP.
1b	Indicates that Congestion Notification Packets (CNP) are sent using the UP value in *CNP_UP_override*.

use_VLAN (1 bit)

Value	Description
0b	The priority field is put into the TOS/DSCP field. Bits 0-5 of each byte are used to map the 64 DSCP priorities.
1b	The priority field is put into a VLAN. Bits 0-2 of each byte are used to map the priority.

11.5.3.29 Query RDMA Features

This WQE is used to query the RDMA features and the firmware version. Firmware returns the list of features up to *buffer_len*. Software is responsible for ensuring that the buffer is large enough.

The *feature_cnt* indicates the number of entries that firmware would return assuming the *buffer_len* is large enough. Software should allocate a reasonably-sized buffer. If needed, software can determine if the buffer is too small by looking at the returned *feature_cnt* field, allocate a larger buffer, and re-issue the WQE.

Table 11-63. Query RDMA Features Descriptor Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	RSVD		
8	[63:0]	RSVD		
16	[63:0]	RSVD		
24	[63] [62:38]	WQE_Valid RSDV	[37:32] [31:0]	OP buffer_len
32	[63:0]	Physical_Buffer_Address		
40	[63:0]	RSVD		
48	[63:0]	RSVD		
56	[63:0]	RSVD		

OP (6 bits) and WQE_Valid (1 bit)

See Section 11.5.3.1.

Physical_Buffer_Address (64 bits)

This field specifies the physical address of the buffer that firmware fills with the RDMA feature information. The buffer must be aligned to an 8-byte boundary.

buffer_len

Length of the buffer provided by software. The minimum length is 8 bytes and must be a multiple of 8 bytes.

Table 11-64. Query RDMA Features Buffer Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0-n	[63:48]	feature_type	[47:0]	feature specific information

feature_type

Enumerated value.

Table 11-65. Query RDMA Feature Types with Specific Information

Feature Type Number	Feature Type Name	Feature Specifics
0	RDMA_FTN_Header (Must always be the first entry in the table.)	[63:48] 0 [47:32] feature_cnt (minimum is 1) [31:16] firmware_major_version [15:0] firmware_minor_version
1	RDMA_FTN_HW_VERSION	[63:48] 1 [47:32] hw_model_used [31:16] hw_major_version [15:0] hw_minor_version
24	RDMA_FTN_ENDPT_TRK	[63:48] 24 [39:1] RSVD [0] 0b = Disabled. 1b = Enabled
26	RDMA_FTN_QSETS_MAX	[63:48] 26 [47:16] RSVD [15:0] qsets_max

hw_model_used (16 bits)

The lower value of the *hw_major_version* of the RNIC hardware and the *hw_major_version* supplied by software at Create CQP time.

hw_major_version, hw_major_version (16 bits each)

See CQP Context description in Section 11.5.2.3.

qsets_max (16 bits)

The maximum number of QSets supported. Each leaf node in the RDMA Work Scheduler refers to a QSet.

11.6 RDMA Functionality

The operations for RDMA via iWARP and RoCEv2 RC are implemented using the context and WQE formats described in the following sections. QP operation, the verbs interface, and system view of the E810 are described in Section 11.4.1. The CQP operations and HMC structures necessary to bring the E810 to a functional state for RDMA are described in Section 11.5.

The maximum number of PE contexts for the device is 256K. These contexts can be allocated in a flexible manner between the functions with a maximum number limit per each function of 256K - 1.

11.6.1 iWARP Q2 Area

iWARP QPs have a memory area that is written under error conditions. The format of this area is listed in Table 11-66. This memory area is reference from the *Q2_Address* variable from the QP context listed in Table 11-67.

Table 11-66. iWARP Q2 Structure Format

Byte Offset	[Bit Range] Field Name
0-55	Outbound_Terminate_Header
56-63	RSVD
64	[63] Sequence_Update_Toggle [62:32] RSVD [31:0] First_Partial_Sequence_Number
72-255	Inbound_Q2_Data

Outbound_Terminate_Header (56 bytes)

This field contains the outbound terminate header used for Modify QP CQP operations when *Terminate_Actions* is set to "Send Terminate Only" or "Send both Terminate and FIN".

First_Partial_Sequence_Number (32 bits)

This field contains the sequence number of the first partial FPDU receive by the E810. This field is defined for partial FPDU support only, and must be ignored during Asynchronous Event (AE) processing. See Section 11.4.3 for more details on partial FPDU support.

Sequence_Update_Toggle (1 bit)

This field toggles each time the *First_Partial_Sequence_Number* is updated by the E810. This field is defined for partial FPDU support only, and must be ignored during Asynchronous Event (AE) processing. See Section 11.4.3 for more details on partial FPDU support.

Inbound_Q2_Data (184 bytes)

This field contains the first 184 bytes of the packet that caused an AE. This could either be an inbound terminate message or the Ethernet packet that had an error.

11.6.2 iWARP QP Context Format

iWARP QP context is used by software to initialize or update the E810 QP context. Create and Modify QP CQP operations (see Section 11.5.3.2) reference this structure.

Table 11-67. iWARP QP Context Structure Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:62] [61:48] [47] [46:42] [41:32] [31] [30] [29] [28] [27:26] [25] [24:22] [21:20] [19]	iwarp_rdmap_ver[3] RSVD Push_Mode_Enable RSVD Push_Page_Index SQ_TPH_en RQ_TPH_en XMIT_TPH_en RCV_TPH_en RSVD DCTCP_enable RSVD pd_index_high err_RQ_index_valid[1]	[18:16] [15] [14] [13:12] [11:10] [9:8] [7] [6] [5] [4] [3] [2] [1:0]	dupack_thresh drop_out_of_order_seg ECN_enable limit RSVD RQ_WQE_Size timestamp RSVD Insert_VLAN_Tag NoNagle[2] IPv4 infiniband_read_en iwarp_ddp_ver[3]
8	[63:0]	SQ_Address		
16	[63:0]	RQ_Address		
24	[63:48] [47:32] [31:24] [23]	Dest_Port_Number Source_Port_Number Traffic_Class_or_TOS[2] avoid_stretch_ack	[22:16] [15:12] [11:8] [7:0]	RSVD SQ_Size RQ_Size Hop_Limit_or_TTL[2]
32	[63:32]	Dest_IP_Address_2	[31:0]	Dest_IP_Address_3
40	[63:32]	Dest_IP_Address_0	[31:0]	Dest_IP_Address_1
48	[63:48] [47:32] [31:30]	ARP_Index VLAN_Tag[2] syn_rst_handling	[29:16] [15:0]	snd_mss RSVD
56	[63:48] [47:44] [43:40] [39:36] [35:32] [31:28]	pd_index RSVD Snd_wscale RSVD Rcv_wscale TCP_state	[27:24] [23] [22] [21] [20] [19:0]	RSVD ignore_tcp_uns_options ignore_tcp_options RSVD wscale Flow_Label[2]
64	[63:0]	RSVD		
72	[63:32]	timestamp_age	[31:0]	timestamp_recent
80	[63:32]	snd_wnd	[31:0]	snd_nxt
88	[63:32]	rcv_wnd[2]	[31:0]	rcv_nxt
96	[63:32]	snd_una	[31:0]	snd_max
104	[63:32]	rtt_var	[31:0]	srtt
112	[63:32]	cwnd	[31:0]	ss_thresh
120	[63:32]	snd_wl2	[31:0]	snd_wl1
128	[63:57] [56:54] [53:48]	Manage Work Scheduler RSVD rexmit_thresh	[47] [46:32] [31:0]	RSVD err_RQ_index max_snd_window
136	[63:51] [50:32]	RSVD RxCmpQueueNum	[31:19] [18:0]	RSVD TxCmpQueueNum
144	[63:8] [7]	Q2_Address RSVD	[6:0]	Statistics_Instance_Index

Table 11-67. iWARP QP Context Structure Format [continued]

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
152	[63:56] [55:48] [47:40] [39:32]	Src_MAC_Address[5] Src_MAC_Address[4] Src_MAC_Address[3] Src_MAC_Address[2]	[31:24] [23:16] [15:8] [7:0]	Src_MAC_Address[1] Src_MAC_Address[0] RSVD last_byte_sent
160	[63:57] [56:48] [47:41] [40:32] [31] [30] [29] [28] [27] [26]	RSVD snd_mrk_offset RSVD rcv_mrk_offset[3] rcv_no_mpa_crc[3] assume_aligned_headers Receive_Markers[3] iWARP_Mode TimelyEnable Use_Statistics_Instance	[25] [24] [23] [22] [21] [20] [19] [18:16] [15:8] [7:0]	PrivilegedEnable FastRegisterEnable BindEnable Send_Markers rdmard_ok rdmawr_rdresp_ok RSVD IRD_Size RSVD ORD_Size
168	[63:0]	QP_Completion_Context		
176	[63:50] [49:32] [31:26]	RSVD Exception_UDA_Queue RSVD	[25:16] [15:8] [7:0]	QS_Handle RQ_TPH_value SQ_TPH_value
184	[63:32]	Local_IP_Address_2	[31:0]	Local_IP_Address_3
192	[63:32]	Local_IP_Address_0	[31:0]	Local_IP_Address_1
200	[63:52] [51:40]	t_high RSVD	[39:32] [31:0]	t_low RSVD
208	[63:0]	RSVD		
216	[63:48] [47:32]	RSVD remote_endpoint_index	[31:0]	RSVD
224-248	[63:0]	RSVD		

1. *err_RQ_index_valid* and *err_RQ_index* are only valid on Modify QP to Terminate or Error.
2. This variable is a cached variable. See *Cached_Variables_Valid* in Modify QP operations for more details on cached context variables.
3. Only valid for iWARP QPs.

Push_Page_Index (10 bits)

This field identifies the push page associated with the QP. For VFs, the push page index is function-relative.

Push_Mode_Enable (1 bit)

This field indicates if push mode is enabled for the QP.

ignore_tcp_options (1 bit)

Specifies that the TCP options should be ignored when processing received TCP headers.

ignore_tcp_uns_options (1 bit)

Specifies that the unsupported TCP options should be ignored when processing received TCP headers.

NoNagle (1 bit)

This field indicates if Nagle algorithm is in use on this connection. If this field is set (1b), the Nagle algorithm is disabled on this connection (RFC 896).

PrivilegedEnable (1 bit)

This bit is used to enable privilege mode on STags.

Value	Description
0b	STag 0 is NOT allowed for a local STag, and an AEQE is generated with privilege error indicated.
1b	A local STag of zero (STag 0) is allowed on this connection, and the *TO* field should be treated as a physical address.

STag 0 is never allowed in an inbound RDMA packet.

FastRegisterEnable (1 bit)

This bit is used to enable fast register opcodes on privilege mode QPs.

Value	Description
0b	A fast register is NOT allowed for this connection, and an AEQE is generated with privilege error indicated.
1b	A fast register is allowed on this connection.

BindEnable (1 bit)

This bit is used to enable memory window bind operations on this QP.

Value	Description
0b	Memory window bind operations are NOT allowed on this connection, and an AEQE is generated with privilege error indicated.
1b	Memory window bind operations are allowed on this connection.

TimelyEnable (1 bit)

This bit enables the TIMELY congestion control algorithm.

Receive_Markers (1 bit)

This field specifies if we are to receive MPA markers in the receive stream.

Value	Description
0b	No markers are expected in the receive stream.
1b	Markers are expected every 512 bytes.

Send_Markers (1 bit)

This field specifies if generating MPA markers in the transmit stream.

Value	Description
0b	Markers are not included in the transmit stream.
1b	Markers are expected every 512 bytes.

rdmard_ok (1 bit)

Value	Description
0b	Inbound iWARP RDMA read requests are disabled, and an AEQE is generated.
1b	Inbound iWARP RDMA read requests are enabled.

rdmawr_rdresp_ok (1 bit)

Value	Description
0b	Inbound iWARP RDMA write requests and inbound RDMA read responses are disabled, and an AEQE is generated.
1b	Inbound iWARP RDMA write requests and inbound RDMA read responses are enabled.

timestamp (1 bit)

This field indicates if timestamp option is in use on this connection.

Value	Description
0b	The timestamp option is not present.
1b	The timestamp option is enabled on this connection (RFC 1323).

wscale (1 bit)

This field indicates if window scale option is in use on this connection.

Value	Description
0b	Windows are not scaled.
1b	The window scale option is enabled on this connection in both directions (RFC 1323). The window is scaled by the *snd_wscale* and *rcv_wscale* variables described later.

dupack_thresh (3 bits)

Specifies the number of dupacks received before starting fast retransmit (1-7). A value of 0 is remapped to a default value of 3.

err_RQ_index_valid (1 bit)

Indicates that the *err_RQ_index* is valid. This setting is allowed only for a Modify QP to Terminate or a Modify QP to Error operation.

err_RQ_index (15 bits)

Software uses *err_RQ_index* to override the index in the RQ during a FlushWQEs CQP operation. This is only valid if *err_RQ_index_valid* is on and it is a Modify QP to Terminate or Modify QP to Error operation.

drop_out_of_order_seg (1 bit)

Value	Description
0b	The out-of-order segment is processed.
1b	If a segment is received that is out-of-order, TRX drops the segment and sends an acknowledgment as required by TCP.

DCTCP_enable (1 bit)

Value	Description
0b	Data Center TCP is disabled.
1b	Data Center TCP is enabled and *ECN_enable* is ignored.

TimelyEnable, and *DCTCP_enable* are mutually exclusive. At most, only one can be set.

ECN_enable (1 bit)

Value	Description
0b	ECN functionality per RFC 3168 is disabled.
1b	The *TOS* field of the IPv4 header or *IP TC* field of the IPv6 header contains the ECN code point (bits 6 and 7 - least significant bits) and the code point in the *IP TC* or *TOS* context field must be set to 01b (normal) or code point 10b (optional).

XMIT_TPH_en (1 bit)

Value	Description
0b	THP is not used for data reads associated with this QP.
1b	TPH is enabled for data reads associated with this QP.

SQ_TPH_en (1 bit)

Value	Description
0b	THP is not used for this resource.
1b	TPH is enabled for the SQ of this QP.

SQ_TPH_value (8 bits)

If *SQ_TPH_en* is set (1b), TPH STag associated with SQ operations is initialized with *SQ_TPH_value*. If *SQ_TPH_en* is clear (0b), this field is ignored.

SQ_Size (4 bits)

This field encodes the maximum size for the WQ. The encoding of the SQ sizes are $4*2^{SQ\_Size}$ in terms of 32-byte quanta of memory. The following value are allowed for *SQ_Size*:

Value	Description
0001b	256 bytes
0010b	512 bytes
0011b	1024 bytes
0100b	2048 bytes
0101b	4096 bytes
0110b	8192 bytes
0111b	16384 bytes
1000b	32768 bytes
1001b	65536 bytes

Value	Description
1010b	131072 bytes
1011b	262144 bytes
1100b	524288 bytes
1101b	1048576 bytes
Note: All other values are reserved.	

Software can only allocate N-1 WQEs on the SQ. Each WQE is variable in size and can consume up to 256 bytes of memory. For more information see Section 11.4.1.5.2. The minimum size for an SQ is four WQEs of the maximum size that is used.

RCV_TPH_en (1 bit)

Value	Description
0b	THP is not used for data placement associated with this QP.
1b	TPH is enabled for data placement associated with this QP.

RQ_TPH_en (1 bit)

Value	Description
0b	THP is not used for this resource.
1b	TPH is enabled for the RQ of this QP.

RQ_TPH_value (8 bits)

If *RQ_TPH_en* is set (1b), TPH STag associated with RQ operations is initialized with *RQ_TPH_value*. If *RQ_TPH_en* is clear (0b), this field is ignored.

RQ_Size (4 bits)

This field encodes the maximum size for the WQ. The encoding of the RQ sizes are $4*2^{RQ\_Size}$ in terms of 32-byte quanta of memory. The following value are allowed for *RQ_Size*:

Value	Description
0001b	256 bytes
0010b	512 bytes
0011b	1024 bytes
0100b	2048 bytes
0101b	4096 bytes
0110b	8192 bytes
0111b	16384 bytes
1000b	32768 bytes
1001b	65536 bytes
1010b	131072 bytes
1011b	262144 bytes
1100b	524288 bytes

Value	Description
1101b	1048576 bytes
Note: All other values are reserved.	

The actual number or WQEs that can be posted to the RQ is the size of the WQ divided by the WQE size determined from *RQ_WQE_Size*. Software can only submit N-1 WQEs to a WQ without processing completions for the WQ without exposing the possibility of a WQ overflow. WQ overflow results in indeterministic behavior for the affected WQ. The minimum size for an RQ is eight WQEs.

RQ_WQE_Size (2 bits)

Specifies the number of 32-byte chunks of memory included with each RQ WQE. The maximum number of additional fragments allowed for an RQ WQE is 13 for a total of 14 fragments and a maximum WQE size of 256 bytes. Valid values are:

Value	Description
00b	32 bytes per WQE (no additional fragments)
01b	64 bytes per WQE (1 or 2 additional fragments)
10b	128 bytes per WQE (3 to 6 additional fragments)
11b	256 bytes per WQE (7 to 13 additional fragments)

Insert_VLAN_Tag (1 bit)

This bit is set to enable VLAN processing on a connection. The tag configured in the *VLAN_Tag* field is used for all processing. If this bit is clear (0b), there is no VLAN insertion or removal performed by the PE. Additional VLAN and priority setting can be configured through the VSI associated with the QP. The most significant portion of the VLAN tag carry the user-specified priority.

Value	Description
0b	Received packets destined for the QP with VLAN tags are sent to LAN queues.
1b	Received packets destined for the QP with mis-matched VLAN or no VLAN tag are sent to LAN queues.

IPv4 (1 bit)

This field indicates if the QP is IPv4 or IPv6.

Value	Description
0b	Indicates IPv6.
1b	Indicates IPv4.

infiniband_read_en (1 bit)

Value	Description
0b	Indicates that the RDMA Read operation is iWARP style.
1b	Indicates that the RDMA Read operation can do InfiniBand style reads.

iwarp_ddp_ver (2 bits)

These bits are used to set and check the DV fields of DDP PDUs for this connection.

iwarp_rdmap_ver (2 bits)

These bits are used to set and check the RV fields of RDMAP PDUs for this connection.

RxCmpQueueNum (19 bits)

This field specifies which of the 512K CQs is used for receive completion notification. The Rx and Tx completions can be mapped to the same CQ or different queues.

TxCmpQueueNum (19 bits)

This field specifies which of the 512K CQs is used for transmit completion notification.The Rx and Tx completions can be mapped to the same CQ or different queues.

SQ_Address (64 bits)

If *Virtual_WQs* bit is clear (0b), this field holds SQ base physical address. It must be aligned to an address divisible by 256 bytes. If *Virtual_WQs* bit is set (1b), this field specifies the first HMC PBLE index of the 1-level page list for the SQ (first *first_pm_pbl_index*).

RQ_Address (64 bits)

If *Virtual_WQs* bit is clear (0b), this field holds RQ base physical address. It must be aligned to an address divisible by 256 bytes. This is the RQ base pointer when this is an iWARP accelerated connection. If *Virtual_WQs* bit is set (1b) and the QP is not associated with a shared RQ, this field specifies the first HMC PBLE index of the 1-level page list for the SQ (first *first_pm_pbl_index*).

Traffic_Class_or_TOS (8 bits)

This field specifies the IPv4 type of service bits (RFC 2474). If these bits represent the IPv4 TOS bits, only the lower 4-bits are valid. These bits are set by software when the connection is created and transmitted in the IP header of all sent datagrams for this connection.

rexmit_thresh (6 bits)

Specifies the number of re-transmissions on this connection that can occur before the AE_LLP_TOO_MANY_RETRIES AE is generated. A value of 0b disables generation of the AE and allows infinite retries.

err_RQ_index_valid (1 bit)

Indicates that the *err_RQ_index* is valid. This setting is allowed only for a Modify QP to Error operation.

err_RQ_index (15 bits)

Software uses *err_RQ_index* to override the index in the RQ during a Flush WQEs CQP operation. This is only valid if *err_RQ_index_valid* is set and it is a Modify QP to Error operation.

avoid_stretch_ack (1 bit)

Value	Description
0b	ACKs can be coalesced if the E810 is busy. In the E810, this works for all cases except when an immediate ACK needs to be generated, in which case the ACK sequence number can advance by more than 2*MSS segments.
1b	An ACK is generated ever 2*MSS sequence numbers from *cur_ack_seq_num* up to current *rcv_nxt* value.

This field is normally clear for iWARP connections.

QS_Handle (10 bits)

This field specifies Tx-Scheduler Queue Set handle associated with the TC for this QP. See Section 8.3.3.4 for more information on scheduler configuration. For VFs, the *QS_Handle* is checked to ensure that the VF issuing the CQP command is associated with the *QS_Handle*.

Exception_UDA_Queue (18 bits)

This field specifies UDA queue that receives partial iWARP FPDUs and TCP/IP packets with the URG bit set. The exception queue MUST use 32-byte RQ WQEs.

limit (2 bits)

This field specifies a limit value for the number of bytes that increase cwnd for each received acknowledgment (as per RFC3465). If an acknowledgment is for less than limit, *cwnd* is advanced by that number of bytes. Generally, this field should be set to 3 (11b) for iWARP connections; other values are for inter-operation with various TCP offload engines:

Value	Description
00b	*cwnd* updated by at most 1*SMSS bytes
01b	*cwnd* updated by at most 2*SMSS bytes.
10b	*cwnd* updated by at most 4*SMSS bytes (experimentation).
11b	*cwnd* updated by the number of bytes Acknowledged (experimentation).

if *limit* = 3 (*cwnd* += *ACK_SEQ* - *snd_una*)

else *cwnd* += min(*ACK_SEQ* - *snd_una*, SMSS<<*limit*)

Hop_Limit_or_TTL (8 bits)

This field specifies the IPv4 Time-To-Live (TTL) parameter in the IP header (RFC 791). It is initialized by software.

Dest_Port_Number (16 bits)

This field specifies the destination TCP port number for the TCP header (RFC 793).

Source_Port_Number (16 bits)

This field specifies the source TCP port number for the TCP header (RFC 793).

Dest_IP_Address_0 (32 bits)

This field specifies bits 127 through 96 of the IPv6 IP Addresses. Reserved when *IPv4* is set (1b). The MSB of this field (Bits 31:24) is the first byte on the wire for this field.

Dest_IP_Address_1 (32 bits)

This field specifies bits 95 through 64 of the IPv6 IP Addresses. Reserved when *IPv4* is set (1b).

Dest_IP_Address_2 (32 bits)

This field specifies bits 63 through 32 of the IPv6 IP Addresses. Reserved when *IPv4* is set (1b).

Dest_IP_Address_3 (32 bits)

This field specifies the 32-bit IPv4 (see RFC 791) or the least significant 32-bits of the IPv6 IP Address. The LSB of this field (Bits 7:0) is the last byte on the wire for this field. For IPv4 Addresses, the MSB (Bits 31:24) of this field is the first byte of the destination IP Address on the Ethernet wire.

Local_IP_Address_0 (32 bits)

This field specifies bits 127 through 96 of the IPv6 IP Addresses. Reserved when *IPv4* is set (1b). The MSB of this field (Bits 31:24) is the first byte on the wire for this field.

Local_IP_Address_1 (32 bits)

This field specifies bits 95 through 64 of the IPv6 IP Addresses. Reserved when *IPv4* is set (1b).

Local_IP_Address_2 (32 bits)

This field specifies bits 63 through 32 of the IPv6 IP Addresses. Reserved when *IPv4* is set (1b).

Local_IP_Address_3 (32 bits)

This field specifies the 32-bit IPv4 (see RFC 791) or the least significant 32-bits of the IPv6 IP Address. The LSB of this field (Bits 7:0) is the last byte on the wire for this field. For IPv4 Addresses, the MSB (Bits 31:24) of this field is the first byte of the destination IP Address on the Ethernet wire.

Src_MAC_Address[5-0] (8 bits each)

Specifies the MAC Address associated with the source IP Address. Index 0 is the LSB and byte 5 is the MSB.

snd_mss (14 bits)

Maximum segment size the sender is allowed to transmit. The E810 *snd_mss* is often set to values smaller than the maximum based on the capabilities of the fabric or connection partner. It can change based on the minimum of the *rcv_mss*, Path MTU or Ethernet MTU. This field needs to be set to a value less any TCP options. Rather than setting the value to 1460, if the *timestamp* option is enabled it would be set to 1448. If VLAN is enabled on this QP, subtract 4. The snd_mss value will be rounded down to the nearest multiple of 4.

ARP_Index (16 bits)

Index into the ARP cache to specify the Ethernet MAC Address to use for this connection. Initialized by software during connection establishment. Allows access to 65536 entry ARP table.

VLAN_Tag (16 bits)

Specifies one of the 4096 VLAN tags for this connection; three bits of priority and one bit canonical format. All bits are valid. The VLAN is in the lower 12 bits of the *VLAN_Tag* field. The upper three bits are priority.

TCP_state (4 bits)

Following are the state definitions (RFC 793):

Value	Description
0000b	NON EXISTENT (software state)
0001b	CLOSED
0010b	LISTEN (software state)
0011b	SYN_SENT (software state)
0100b	SYN_RECEIVED (software state)
0101b	ESTABLISHED
0110b	CLOSE_WAIT
0111b	FIN_WAIT_1
1000b	CLOSING (software state)
1001b	LAST_ACK (software state)
1010b	FIN_WAIT_2
1011b	TIME_WAIT (software state)
Note: All other values are reserved.	

Flow Label (20 bits)

This field specifies the IPv6 *Flow Label* field contents (RFC 2460). This field is set by software when a connection is offloaded and can be changed during operation.

pd_index (16 bits) and pd_index_high (2 bits)

Protection domain for this context. This specifies which one of the 256K iWARP protection domains this connection context belongs to. There are no reserved index values.

Snd_wscale (4 bits)

This field specifies the shift count used on the advertised window size received from the other end to obtain the real 32-bit Advertised Window Size (AWS). A value of 0 selects no window scaling. A value of 14 supports a maximum window of 1,073,725,440 bytes [(65536 x 16384) - 1) (RFC 1323]. Software initializes this field from the value supplied during connection establishment (such as SYN segment). Valid values are 0-14, supporting windows up to 1 GB -1. A value of 15 is rolled back to a value of 14.

Rcv_wscale (4 bits)

This field specifies the shift count used every time TCP sends a segment to determine the window size to report. The internal 32-bit window size is right shifted by this field to give the value to place in the TCP header. A value of 0 selects no window scaling. A value of 15 supports a maximum window of 1,073,725,440 bytes ((65536 x 16384) - 1) (RFC 1323). Software initializes this field from the value it supplied during connection establishment (like on the SYN segment) based on the receive buffer size. Valid values are 0-14, supporting a receive window up to 1 GB -1. A value of 15 is rolled back to a value of 14.

timestamp_recent (32 bits)

This field is updated when a segment arrives that includes the expected segment number (RFC 1323). This field is returned in the Echo Reply field of a timestamp.

timestamp_age (32 bits)

This field records the value from *tcp_now* (500 ms timer) the last time *ts_recent* was copied from a receive segment (RFC 1323). This field is required for TCP to perform PAWS.

snd_nxt (32 bits)

The next sequence number that TCP sends in a transmitted packet (RFC 793).

snd_wnd (32 bits)

This field specifies the senders Advertised Window Size (AWS). This field is the 16-bit receive window shifted by *snd_wscale* to come up with the AWS (RFC 793). This is a window our partner advertised that limits our transmit operations.

rcv_nxt (32 bits)

This is the next expected receive sequence number. It is also referred to as the left hand receive pointer (RFC 793).

rcv_wnd (32 bits)

This field specifies the receivers AWS. The AWS is flow control imposed by the receiver. This field is used to generate the 16-bit window size reported in the TCP header, shifted by the receive window scale (*rcv_wscale*). This is the value advertised to the sender (RFC 793).

snd_max (32 bits)

This sequence number points to the maximum send sequence number that has been transmitted. This field sets the upper bound of valid ACK when processing a re-transmission timeout (RFC 793). On connection offload this field is normally set to the same sequence number as *snd_nxt* unless a retransmission was in progress when the connection was offloaded.

snd_una (32 bits)

This sequence number points to the oldest unacknowledged sequence number for this connection (RFC 793).

srtt (32 bits)

This field is the smoothed round-trip time (rtt) << 3 (RFC 793 and RFC 2988) in microseconds. If this field is set to 0b on connection offload then the hardware calculates the value.

rtt_var (32 bits)

Round-trip variation (RTTVAR in RFC 2988) in microseconds. This field MUST never be set to 0b if *srtt* is also set to 0b. Make this a non-zero value if *srtt* is 0b.

ss_thresh (32 bits)

Slow start threshold (RFC 2581).

cwnd (32 bits)

This field is the sender congestion widow. It gets incremented by MSS when an ACK arrives. Software should initialize this field to 1b segment size (in bytes). If a duplicate ACK arrives, this is set to min (*cwnd*, *rcv_wnd*) but at least two segments. On a transport timeout this is set to 1b segment (RFC 2581).

snd_wl1 (32 bits)

Segment sequence number used for last send window update (RFC 793).

snd_wl2 (32 bits)

Segment acknowledgment used for last send window update (RFC 793).

max_snd_window (32 bits)

Indicates the largest send window advertised by the remote peer. Used in step 3 of the sender side SWS algorithm. If at least half of the largest window seen so far is available, send a segment.

Q2_Address (56 bits)

This field points to a buffer in host memory that is used to report a terminate message received on this QP. This pointer is the high 56 bits of the address. The low eight bits are implied as zero (thus the buffer is aligned on a 256-byte boundary). The field can be located at a fixed offset based on some other pointer (receive base pointer); it only points to a one entry queue to handle errors (terminate requests).

last_byte_sent (8 bits)

This is the last byte transmitted on the wire. It is only used during window probes. During modified iWARP only window probes, the E810 transmits the last acknowledged data byte on the wire. This corresponds to the last byte sent because a zero window condition is defined as all bytes acknowledged but no window to continue transmitting. Software should initialize it to the last byte sent before the connect was offloaded the connection to the hardware, so if a streaming mode message had been sent, it is the last byte sent of the last MPA message sent. If after a SYN/ACK handshake then initialize this field to 0b.

rcv_no_mpa_crc (1 bit)

This field specifies if we are to check MPA CRCs in the receive stream.

Value	Description
0b	All inbound FPDUs are checked by the E810.
1b	No MPA CRC checks are performed on FPDUs in the receive stream.

assume_aligned_headers (1 bit)

This field is set by software during connection setup if it knows the other adapter is a E810 that does not generate unaligned headers. This enables the E810 to place all segments that arrive if it passes basic header checks even if the packet was received out of order. Markers are not examined.

iWARP_Mode (1 bit)

This field sets the mode of operation when RQ empty conditions are encountered.

Value	Description
0b	The connection is placed in the terminate state and an AE is generated.
1b	The offending packet is dropped and the peer re-transmits the packet in the hope that the empty condition is resolved by the time the re-transmission occurs.

IRD_Size (3 bits)

This field specifies the number of inbound RDMA resources available for this connection (Q1). Valid settings are:

Value	Description
000b	4 WQEs
001b	16 WQEs
010b	64 WQEs
011b	128 WQEs
100b	256 WQEs
101b	Reserved (defaults to 4 WQEs)
110b	Reserved (defaults to 4 WQEs)
111b	Reserved (defaults to 4 WQEs)

When advertising resources to the other side, only half the queue is available. For example, if *IRD_Size* is set to 64 entries, the ORD size for the connection partner should be set to 32 or less.

ORD_Size (8 bits)

This field specifies the number of outbound RDMA resources available for this connection. Up to 255 outbound RDMA read requests are supported. This field is not encoded because a partner might support a different number of RDMA read requests than on transmit (such as 1, or 10, or 64, and so on).

snd_mrk_offset (9 bits)

This field specifies the offset to the MPA marker in the transmit stream of bytes. Subsequent markers are located every 512 bytes from this location. So modulo math can be used to place the markers in the transmitted byte stream. Software initializes this field upon connection setup.

rcv_mrk_offset (9 bits)

This field specifies the offset to the MPA marker in the receive stream of bytes. Subsequent markers are located every 512 bytes from this location. So modulo math can be used to extract the markers in the receive byte stream. Software initializes this field upon connection setup.

QP_Completion_Context (64 bits)

This field is reported in CQEs and also in AEQEs.

Use_Statistics_Instance (1 bit)

This field indicates if the default per Private Memory Function statistics are used or if one of the additional RDMA statistics instances are used for this QP.

Value	Description
0b	The default statistics are used.
1b	The statistics instance indicated by the *Statistics_Instance_Index* field are used.

Statistics_Instance_Index (7 bits)

This field specifies which of the additional RDMA statistics indexes are used if *Use_Statistics_Instance* is set to 1b. This field is ignored if *Use_Statistics_Instance* is set to 0b.

t_high (12 bits)

The high RTT threshold in microseconds for TIMELY. If the current RTT is higher than this threshold, the congestion window is multiplicatively decreased. If zero, the default value of 500 microseconds is used. This value applies only if *TimelyEnable* is set (1b).

t_low (8 bits)

The low RTT threshold in microseconds for TIMELY. If the current RTT is less than this threshold, the congestion window will be additively increased. If zero, the default value of 50 microseconds is used. This value applies only if *TimelyEnable* is set (1b).

syn_rst_handling (2 bits)

This field indicates how syns and resets are handled.

Value	Description
00b	Hardware uses RFC 5961 syn/rst handling.
01b	Hardware uses RFC 793 syn/rst handling.
10b	Hardware passes syn/rst to firmware which uses RFC 5961 syn/rst handling.
11b	Hardware passes syn/rst to firmware which uses RFC793 syn/rst handling.

remote_endpoint_index (16 bits)

Index in the local table of the remote endpoint. This field is only valid if the *remote_endpoint_trk_en* bit is set (see Section 11.5.2.3). This field is experimental.

RTOmin(7)

Provides a minimum floor for the TCP retransmission timeout value. This field is defined in 10 ms units and has a maximum value of 100, which equates to a 1 s min_rto. Any value greater than 100 and a value of 0 are treated as if a value of 100 was provided.

11.6.3 RoCEv2 QP Context Format

RoCEv2 QP context is used by software to initialize or update the E810 QP context. Create and Modify QP CQP operations (see Section 11.5.3.2) reference this structure for RoCEv2 QPs.

Table 11-68. RoCEv2 QP Context Structure Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:60] [59:48] [47] [46:42] [41:32] [31] [30] [29] [28] [27:26] [25] [24:22]	roce_tver RSVD Push_Mode_Enable RSVD Push_Page_Index SQ_TPH_en RQ_TPH_en XMIT_TPH_en RCV_TPH_en RSVD DCTCP_enable RSVD	[21:20] [19] [18:15] [14] [13:10] [9:8] [7] [6] [5] [4] [3] [2:0]	pd_index_high err_RQ_index_valid[1] RSVD ECN_enable RSVD RQ_WQE_Size RSVD is_QP1 Insert_VLAN_Tag RSVD IPv4 RSVD
8	[63:0]	SQ_Address		
16	[63:0]	RQ_Address		
24	[63:48] [47:32] [31:24] [23:16]	Dest_Port_Number Source_Port_Number Traffic_Class_or_TOS RSVD	[15:12] [11:8] [7:0]	SQ_Size RQ_Size Hop_Limit_or_TTL
32	[63:32]	Dest_IP_Address_2	[31:0]	Dest_IP_Address_3
40	[63:32]	Dest_IP_Address_0	[31:0]	Dest_IP_Address_1
48	[63:48] [47:32] [31:30]	ARP_Index VLAN_Tag RSVD	[29:16] [15:0]	mtu RSVD
56	[63:48] [47:32] [31:25]	pd_index p_key RSVD	[24:20] [19:0]	ack_credits Flow_Label
64	[63:32] [31:24]	q_key RSVD	[23:0]	Dest_QPN
72	[63:0]	RSVD		
80	[63:56] [55:32]	RSVD lsn	[31:24] [23:0]	RSVD psn_nxt
88	[63:24]	RSVD	[23:0]	epsn
96	[63:56] [55:32]	RSVD psn_una	[31:24] [23:0]	RSVD psn_max
104	[63:0]	RSVD		
112	[63:56] [55:32]	RSVD cwnd	[31:0]	RSVD
120	[63:0]	RSVD		
128	[63:57] [56:54] [53:48]	RSVD rnr_nak_thresh rexmit_thresh	[47] [46:32] [31:0]	RSVD err_RQ_index RSVD
136	[63:51] [50:32]	RSVD RxCmpQueueNum	[31:19] [18:0]	RSVD TxCmpQueueNum
144	[63:7]	RSVD	[6:0]	Statistics_Instance_Index

Table 11-68. RoCEv2 QP Context Structure Format [continued]

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
152	[63:56] [55:48] [47:40] [39:32]	Src_MAC_Address[5] Src_MAC_Address[4] Src_MAC_Address[3] Src_MAC_Address[2]	[31:24] [23:16] [15:0]	Src_MAC_Address[1] Src_MAC_Address[0] RSVD
160	[63:32] [31] [30:29] [28] [27] [26] [25] [24]	RSVD rcv_no_icrc RSVD fw_cc_enable TimelyEnable Use_Statistics_Instance PrivilegedEnable FastRegisterEnable	[23] [22] [21] [20] [19] [18:16] [15:8] [7:0]	BindEnable DCQCNEnable rdmard_ok rdmawr_rdresp_ok UDPrivCQEnable IRD_Size RSVD ORD_Size
168	[63:0]	QP_Completion_Context		
176	[63:26] [25:16]	RSVD QS_Handle	[15:8] [7:0]	RQ_TPH_value SQ_TPH_value
184	[63:32]	Local_IP_Address_2	[31:0]	Local_IP_Address_3
192	[63:32]	Local_IP_Address_0	[31:0]	Local_IP_Address_1
200	[63:52] [51:40]	t_high RSVD	[39:32] [31:0]	t_low RSVD
208	[63:0]	RSVD		
216	[63:48] [47:32]	RSVD remote_endpoint_index	[31:0]	RSVD
224-248	[63:0]	RSVD		

1. *err_RQ_index_valid* and *err_RQ_index* are only valid on Modify QP to Error.

For the following field definitions, see Section 11.6.2

- Push_Mode_Enable (1 bit)
- Push_Page_Index (10 bits)
- SQ_TPH_en (1 bit)
- RQ_TPH_en (1 bit)
- XMIT_TPH_en (1 bit)
- RCV_TPH_en (1 bit)
- DCTCP_enable (1 bit)
- pd_index_high (2 bits)
- ECN_enable (1 bit)
- RQ_WQE_Size (2 bits)
- Insert_VLAN_Tag (1 bit)
- IPv4 (1 bit)
- SQ_Address (64 bits)
- RQ_Address (64 bits)
- Dest_Port_Number (16 bits)
- Source_Port_Number (16 bits)
- Traffic_Class_or_TOS (8 bits)
- Source_MAC_Address_Index (6 bits)
- SQ_Size (4 bits)
- RQ_Size (4 bits)
- err_RQ_index (15 bits)
- pd_index (15 bits)
- Flow_Label (20 bits)
- RxCmpQueueNum (19 bits)
- TxCmpQueueNum (19 bits)
- Statistic_Instance_Index (4 bits)
- Use_Statistics_Instance (1 bit)
- PrivilegedEnable (1 bit)
- FastRegisterEnable (1 bit)
- BindEnable (1 bit)
- TimelyEnable (1 bit)
- rdmard_ok (1 bit)
- rdmawr_rdresp_ok (1 bit)
- IRD_Size (3 bits)
- ORD_Size (8 bits)
- QP_Completion_Context (64 bits)
- Exception_UDA_Queue (18 bits)
- QS_Handle (10 bits)
- RQ_TPH_value (8 bits)
- SQ_TPH_value (8 bits)

- Hop_Limit_or_TTL (8 bits)
- Dest_IP_Address_0 (32 bits)
- Dest_IP_Address_1 (32 bits)
- Dest_IP_Address_2 (32 bits)
- Dest_IP_Address_3 (32 bits)
- ARP_Index (16 bits)
- VLAN_Tag (16 bits)
- err_RQ_index_valid (1 bit)
- Local_IP_Address_0 (32 bits)
- Local_IP_Address_1 (32 bits)
- Local_IP_Address_2 (32 bits)
- Local_IP_Address_3 (32 bits)
- t_high (12 bits)
- t_low (8 bits)
- remote_endpoint_index (16 bits)
-

roce_tver (4 bits)

The protocol engine uses this value when generating the BTH and compares this value to received BTH headers. The only valid value is 0.

p_key (16 bits)

This field is the Partition Key. It is required for RC and UD QPs.

q_key (32 bits)

This field is the Queue Key. It is used for UD flows. This value is used as the *q_key* if the value in the sending request has the most significant bit set.

ack_credits (5 bits)

This is used to set the initial encoded ACK credit syndrome field in ACKs issued by this node for this connection. The only value defined is 11111b, which disables participation in credit management for this connection on this node.

Dest_QPN (24 bits)

This field is the destination QP number. It is used for RC QPs.

psn_nxt (24 bits)

This field is the send packet sequence number. It is set to the initial psn value for the QP. It should be set to a random value.

epsn (24 bits)

This field is the expected incoming (receive) packet sequence number. It is required for RC connections. The Starting PSN is received in the REQ/REP headers during connection setup.

psn_una (24 bits)

This field is the PSN of the unacknowledged request. On QP create, it should be set to *psn_nxt*.

psn_max (24 bits)

This field is the high watermark of Tx requests. It should be set to the same value as *psn_nxt* at QP create time.

mtu (14 bits)

The maximum payload size. The only supported values are 256, 512, 1024, 2048,and 4096.

is_QP1 (1 bit)

Indicates that the QP is used for RoCEv2 Management Datagram (MAD) processing. This bit is not valid for RC connections. This bit is intended to be used for VMs (not PFs/VFs which have their own QP1).

UDPrivCQEnable (1 bit)

Indicates that the additional values, SMAC and VLAN, are in the UD completion. This bit is not valid for RC connections.

DCQCNEnable (1 bit)

Enables the DCQCN algorithm for RoCEv2. *DCQCNEnable*, *TimelyEnable*, and *DCTCP_enable* are mutually exclusive. At most, only one can be set. When this bit is set (1b), Data Center TCP is enabled and *ECN_enable* is ignored. *TimelyEnable*, and *DCTCP_enable* are mutually exclusive. At most, only one can be set.

rcv_no_icrc (1 bit)

This field specifies if whether to check ICRCs in the receive stream.

Value	Description
0b	All inbound ICRCs are checked.
1b	No ICRC checks are performed in the receive stream.

fw_cc_enable (1 bit)

When *fw_cc_enable* is set (1b), all Congestion Notification Packets (CNPs) are forwarded to firmware. Firmware handles the packet and updates the congestion window. When *fw_cc_enable* is set (1b), *DCQCNEnable* and *TimelyEnable* are ignored.

Src_MAC_Address[5-0] (8 bits each)

Specifies the MAC Address associated with the source IP Address. Index 0 is the LSB and byte 5 is the MSB.

cwnd (24 bits)

Initial congestion window in terms of packets.

lsn (24 bits)

Initial setting for the Limit Sequence Number (LSN). LSN limits the number of messages that can be outstanding at one time. Logically, LSN = MSN + credit count.

rexmit_thresh (6 bits)

Specifies the number of re-transmissions on this connection that can occur before the AE_LLP_TOO_MANY_RETRIES AE is generated. A value of 0b disables generation of the AE and allows infinite retries.

rnr_nak_thresh (3 bits)

Specifies the number of rnr_naks on this connection that can occur before the AE_LLP_TOO_MANY_RETRIES AE is generated. A value of 0 disables generation of the AE and allows infinite retries.

11.6.4 RDMA QP Completion Codes

RDMA QP errors are typically reported via AEs with the E810. After fielding the AEs, software might issue the flush WQES CQP operation (see Section 11.5.3.17) to complete any pending operations. Table 11-69 lists the completion codes reported by the E810 in a CQ entry (Table 11-67). Also note that host software can report any completion code necessary using the flush WQEs operation instead of the codes listed in Table 11-69.

Table 11-69. RDMA QP Error Codes

Major Error Code	Minor Error Code	Completion Reason	Description
0x0001	0x0001	WQE Flushed	The WQE has been flushed due to a ModifyQP state transition.

11.6.5 RDMA CQ Entry Formats

CQ operation for RDMA is described in Section 11.4.1.3. CQs are manipulated through CQP operations. See Section 11.5.3.3 for further details.

When *Avoid_Memory_Conflicts* is not set (0b), the CQE is normally 32 bytes. However, some completions require additional information. In this case, the *Extended_CQE* bit is set (1b) and an additional 32 bytes is added just for that CQE.

When *Avoid_Memory_Conflicts* is set (1b), the CQE is always 64 bytes. If the *Extended_CQE* bit is not set (0b), and the second half of the CQE is all zeros. If the *Extended_CQE* bit is set (1b), the second half of the CQE is described in Table 11-71.

Table 11-70. RDMA CQ Entry Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:32]	TCP_or_Packet_Sequence_Number_or_RTT	[31:0]	Payload_Length
8	[63:0]	QP_Completion_Context		
16	[63:50] [49:32]	RSVD QP_ID	[31:0]	Invalidated_STag_or_L/RKey
24	[63] [62] [61:56] [55] [54] [53] [52]	CQE_Valid SQ OP Error Solicited_Event (RDMA RQ only) STag_or_L/RKey (RDMA RQ only) RSVD	[51] [50] [49:47] [46:32] [31:16] [15:0]	Push_Dropped (RDMA SQ only) Extended_CQE RSVD WQ_Desc_Index Major_Error_Code Minor_Error_Code

Table 11-71. Extended CQ Entry Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
32	[63:56] [55:48] [47:40] [39:32]	Immediate_Data Byte[7] Immediate_Data Byte[6] Immediate_Data Byte[5] Immediate_Data Byte[4]	[31:24] [23:16] [15:8] [7:0]	Immediate_Data Byte[3] Immediate_Data Byte[2] Immediate_Data Byte[1] Immediate_Data Byte[0]
40	[63:0]	RSVD		

Table 11-71. Extended CQ Entry Format [continued]

Byte Offset	[Bit Range]	Field Name		
48	[63:0]	RSVD		
56	[63] [62] [61]	CQE_Valid2 Immediate_Data_valid UD_smac_valid	[60] [59:0]	UD_VLAN_Tag_valid RSVD

OP (6 bits)

This field reports the opcode from the operation associated with the CQE.

Extended_CQE (1 bit)

When the *Extended_CQE* bit is set (1b), the CQE is extended to 64 bytes. The second half of the CQE format is shown in Table 11-71. When the *Extended_CQE* is valid, software must ensure that the *CQE_Valid2* flag has the right polarity. If it is not valid, then the CQE is not complete and software cannot yet process the CQE.

WQ_Desc_Index (15 bits)

WQ is sliced up into 32-byte descriptor quanta. Every WQE must start with a 32-byte descriptor on a 32-byte boundary. *WQ_Desc_Index* reports the 32-byte quanta index of the WQE associated with the completion.

CQE_Valid (1 bit)

The *CQE_Valid* bit for CQE indicates that a CQE is ready to be processed. The polarity of the *Valid* bit changes each time the CQ wraps from the last entry back to the first entry. This change in polarity reduces software overhead by avoiding the need to clear the *Valid* bit once software has processed a valid CQE.

Software is responsible to clear (set to 0b) all memory in a CQ initially at CQ creation. The first iteration (and subsequent odd numbered iterations) through the CQ, the E810 sets the Valid bit to 1b when it writes a new CQE. For the second iteration (and all even numbered iterations) through the CQ, the E810 sets the valid bit to 0b when it writes an new CQE.

Error (1 bit)

Value	Description
0b	No error.
1b	An error occurred when processing the WQE associated with this CQE and that the *Error_Code* field is valid.

STag_or_L/RKey (1 bit)

Indicates if the *Invalidated_STag_or_L/RKey* field has valid contents.

Value	Description
0b	Ignore *Invalidated_STag_or_L/RKey*.
1b	An STag or L/RKey was invalidated and *Invalidated_STag_or_L/RKey* field is valid.

Solicited_Event (1 bit)

Valid for receive only. Value for SQ completions is undefined.

Value	Description
0b	SE bit not set in received packet.
1b	SE bit set in received packet.

SQ (1 bit)

Value	Description
0b	RQ
1b	SQ

Push_Dropped (1 bit)

Valid only when SQ is set.

Value	Description
0b	Push operations are being processed successfully.
1b	A recent push mode operation has been dropped by the E810. Software should refrain from submitting additional push mode operations until the SQ has encountered an empty condition.

Payload_Length (32 bits)

Total payload length of the completed message. This field is only valid for RQ WQE completions.

TCP_or_Packet_Sequence_Number_or_RTT (32 bits)

For iWARP, this field reports the TCP sequence number associated with the work request specified by the completion. For SQ-related CQEs, RTT is reported instead of the TCP Sequence number if the SQ WQE has the *ReportRTT* bit set.

For a RoCEv2 SQ completions, this field is the Packet Sequence Number or RTT.

QP_Completion_Context (64 bits)

Completion context pointer. This field is transferred to the CQE from QP context.

QP_ID (18 bits)

QP associated with the completed message.

Invalidated_Stag_or_L/RKey (32 bits)

Any STag or L/RKey that was invalidated by the completed message. Only valid if the *STag or L/RKey* bit is also set. If the *STag or L/RKey* bit is clear, *Invalidated_Stag or L/RKey* must be ignored. This is valid only for RC QPs.

Major_Error_Code (16 bits)

Valid if the *Error* is set (1b). See Table 11-69 for defined values. Software can also report any value via the Flush WQEs CQP operation (see Section 11.5.3.17).

Minor_Error_Code (16 bits)

Valid if *Error* is set (1b). See Table 11-69 for defined values. Software can also report any value via the Flush WQEs CQP operation (see Section 11.5.3.17).

The following items are in the extended CQE:

CQE_Valid2 (1 bit)

The *CQE_Valid2* bit must match the setting for a valid CQE in the CQ. Normally the value of the *CQE_Valid2* bit will be the same as the *CQE_Valid* bit in this completion. However, there is one case where it is different: if the completion starts in the last entry of the CQ, the second half of the CQE wraps to the start of the CQ so the *CQE_Valid2* bit flips to match the *CQE_Valid* setting for the next pass of the CQ.

Immediate_Data_valid (1 bit)

This bit indicates that immediate data was received.

Immediate_Data (8 bytes)

For iWARP, the *Immediate_Data* field is all eight bytes, which are in bytes 0-7 of the *Immediate_Data* field.

For RoCEv2, the *Immediate_Data* field is four bytes, which are in bytes 0-3 of the *Immediate_Data* field, and bytes 4-7 are ignored.

UD_smac_valid (1 bit)

Must be 0b for RC QPs. See Section 11.7.1.1 for a description of UD CQEs.

UD_VLAN_Tag_valid (1 bit)

Must be 0b for RC QPs. See Section 11.7.1.1 for a description of UD CQEs.

11.6.6 RDMA Descriptor Formats

11.6.6.1 RDMA SQ Descriptors

The following WQE formats are used in conjunction with RDMA QPs. Operations that are supported for RDMA QPs are the listed in Table 11-72.

Table 11-72. RDMA QP Operations

Operation Code	Operation Name	Section Reference
0x00	RDMA Write	11.6.6.1.5
0x01	RDMA Read	11.6.6.1.7
0x02	Reserved	N/A
0x03	Send	11.6.6.1.3
0x04	Send with Invalidate	
0x05	Send with Solicited Event	
0x06	Send with Solicited Event and Invalidate	
0x07	Reserved	N/A
0x08	Memory Window Bind	11.6.6.1.8
0x09	Fast Register Memory Region	11.6.6.1.10
0x0A	Local Invalidate STag	11.6.6.1.9
0x0B	RDMA Read with Local Invalidate	11.6.6.1.7
0x0C	NOP	11.6.6.1.2
0x0D	RDMA Write with Solicited Event	11.6.6.1.5
0x0E-0x2F	Reserved	N/A
0x30	Connection Established	11.6.6.1.11
0x31-0x3F	Reserved	N/A

11.6.6.1.1 Common RDMA SQ Descriptor Format Fields

The basic RDMA WQE is a 32-byte structure that is broken up into 64-bit (8-byte) words. The placement of these fields (when they apply) within WQE are common among all CQP WQEs and also RDMA WQEs.

Table 11-73 lists the basic structure of a QP WQE including the common fields. The definition of fields marked as "Operation Code Dependent" vary from operation-to-operation and are detailed in subsequent sections. Fields marked as "Reserved" must be set to 0b or undesired behavior related to the specific QP associated with the WQE might occur. Additionally, RDMA WQEs can optionally include additional descriptors to enable larger WQEs to be created. The additional descriptor format is listed in Table 11-74 and Table 11-2.

Table 11-73. RDMA Common WQE Fields

Byte Offset	[Bit Range] Field Name			
0	[63:0]	Operation Code Dependent		
8	[63:0]	Operation Code Dependent		
16	[63:0]	Operation Code Dependent		
24	[63]	WQE_Valid	[55:48]	Operation Code Dependent (Inline_Data Length)
	[62]	Signaled_Compeltion	[47]	Operation Code Dependent (Immed_Data_Flag)
	[61]	Local_Fence	[46]	Operation Code Dependent (ReportRTT)
	[60]	Read_Fence	[45:42]	RSVD
	[59]	Operation Code Dependent (wait_for_rcvFPDU)	[41:38]	AdditionalFragmentCount
	[58]	Operation Code Dependent (Streaming_Mode)	[37:32]	OP
	[57]	Operation Code Dependent (Inline_Data_Flag)	[31:0]	Operation Code Dependent
	[56]	Push_WQE		

OP (6 bits)

RDMA operation code. See Table 11-72 for the specific values.

WQE_Valid (1 bit)

The *WQE_Valid* bit for Work Queue Entries (WQE) indicates that a WQE is ready to be processed by the E810. The polarity of the *Valid* bit changes each time the WQ wraps from the last entry back to the first entry. This change in polarity reduces software overhead associated with the need to clear a *Valid* bit and also to enable the E810 to read ahead in the WQ to reduce the need for doorbell rings. See Section 11.4.1.5.2 for more information on submitting work to a QP with the E810.

Software is responsible to clear (set to 0b) all memory in a WQ initially at QP creation. The first iteration (and subsequent odd numbered iterations) through the WQ, software sets the *Valid* bit to 1b when it writes a new WQE. For the second iteration (and all even numbered iterations) through the WQ, software sets the *Valid* bit to 0b when it writes an new WQE.

AdditionalFragmentCount (4 bits)

The maximum number of fragments is 14. In the WQE, the *AdditionalFragmentCount* specifies the number of additional fragment descriptors that are valid for this WQE. When there is no Immediate Data, the first fragment descriptor is in the first quanta, so the max for *AdditionalFragmentCount* is 13. When Immediate Data is present, it consumes the initial fragment, so the max *AdditionalFragmentCount* is 14.

Push_WQE (1 bit)

Indicates that the WQE was pushed to the push page associated with the QP. If this bit is set and the E810 dropped the push operation, the *Push_Dropped* bit is set in the next CQE following this WQE to indicate to software to stop pushing WQEs until an empty SQ condition has been observed. Using this rule prevents additional CPU and PCI bus bandwidth from being consumed when the Ethernet fabric is congested or the E810 is temporarily unable to process push mode operations.

Signaled_Completion (1 bit)

Value	Description
0b	Do not generate a CQE for this message unless there is an error associated with the WQE.
1b	Generate a CQE for this WQE. A CEQE might or might not be generated depending on the state of the CQ.

Local_Fence (1 bit)

This bit set (1b) signifies that the current WQE MUST NOT start until all prior SQ WQEs on the QP completed.

ReportRTT (1 bit)

This bit set (1b) indicates that RTT will be reported in the CQE instead of TCP or Packet Sequence Number. This is the latest SRTT (Smoothed Round Trip Time) according to RFC 6298.

Read_Fence (1 bit)

This bit set (1b) signifies that the current WQE MUST NOT start until any outstanding RDMA read requests on the SQ complete.

11.6.6.1.1.1 Fragment Descriptor Format

Table 11-74. RDMA Fragment Descriptor Format

Byte Offset	[Bit Range] Field Name			
0	[63:0]	Tagged_Offset		
8	[63] [62:32]	Fragment_Valid Fragment_Length	[31:0]	STag

Tagged_Offset (64 bits)

The tagged offset (relative to the STag) associated with the data described by the additional fragment descriptor.

STag (32 bits)

STag associated with the data described by the additional fragment descriptor.

Fragment_Valid (1 bit)

Indicates the fragment is valid. A fragment is valid when this bit matches the *WQE_Valid* bit.

In the additional 32-byte descriptor quantas, when only one fragment is used it is still necessary to set the *Fragment_Valid* bit in both fragments. The second fragment in the 32-byte descriptor quanta will not be used since the *AdditionalFragmentCount* did not include it.

Fragment_Length (31 bits)

Length in bytes of the data described by the additional fragment descriptor.

11.6.6.1.1.2 Inline Data Format

Whether or not the WQE includes Immediate Data, the first eight bytes of Inline Data begin in the first fragment (offset 8 in the WQE). If the Inline Data exceeds eight bytes, the data continues in the next 32-byte descriptor quanta(s).

Table 11-75. Inline Data Format for the First 32-Byte Descriptor Quanta in a WQE

Byte Offset	[Bit Range]	Field Name
0	[63:0]	Immed_Data or RSVD
8	[63:0]	Inline_Data
16	[63:0]	Operation Dependent
24	[63:0]	Opcode, flags, and so on

The format of each additional 32-byte descriptor quanta follows:

Table 11-76. Inline Data Format for the Additional 32-Byte Descriptor Quanta(s) in a WQE

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	Inline_Data		
8	[63:0]	Inline_Data		
16	[63:0]	Inline_Data		
24	[63] [62:56]	Inline_Valid RSVD	[55:0]	Inline_Data

The maximum inline data size is 224 bytes.

The *Inline_Valid* flag must be set in every 32-byte descriptor quanta that has inline data. The *Inline_Valid* flag must match the *WQE_Valid* flag.

11.6.6.1.2 SQ WQE Format - NOP

Table 11-77. RDMA SQ NOP WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	RSVD		
8	[63:0]	RSVD		
16	[63:0]	RSVD		
24	[63] [62] [61] [60] [59:57]	WQE_Valid Signaled_Completion Local_Fence Read_Fence RSVD	[56] [55:42] [41:38] [37:32] [31:0]	Push_WQE RSVD AdditionalFragmentCount OP RSVD

For the following field definitions, see Section 11.6.6.1.1.[1]

- WQE_Valid (1 bit)
- Signaled_Completion (1 bit)
- Local_Fence (1 bit)
- Read_Fence (1 bit)
- Push_WQE (1 bit)
- AdditionalFragmentCount (4 bits)[1]

OP (6 bits)

This WQE format is valid for the NOP operation. See Section 11.6.6.1.1 for the associated opcode values.

11.6.6.1.3 SQ WQE Format - Send

Table 11-78. RDMA SQ Send WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	Tagged_Offset		
8	[63] [62:32]	Fragment_Valid Fragment_Length	[31:0]	STag
16	[63:0]	RSVD		
24	[63] [62] [61] [60] [59] [58] [57] [56]	WQE_Valid Signaled_Completion Local_Fence Read_Fence wait_for_rcvFPDU Streaming_Mode Inline_Data_Flag Push_WQE	[55:48] [47] [46] [45:42] [41:38] [37:32] [31:0]	RSVD Immed_Data_Flag ReportRTT RSVD AdditionalFragmentCount OP Remote_Invalidate_STag

For the following field definitions, see Section 11.6.6.1.1.

- WQE_Valid (1 bit)
- Signaled_Completion (1 bit)
- Local_Fence (1 bit)
- Read_Fence (1 bit)
- Push_WQE (1 bit)
- ReportRTT (1 bit)
- AdditionalFragmentCount (4 bits)

1. The *AdditionalFragmentCount* can be used instead of placing multiple individual NOPs.

OP (6 bits)

This WQE format is valid for the "Send", "Send with Invalidate", "Send with Solicited Event", and "Send with Solicited Event and Invalidate", as well as the versions of these operations that include inline data. See Section 11.6.6.1.1 for the associated opcode values.

Streaming_Mode (1 bit)

Indicates that this WQE contains data that is to be sent in streaming mode (normal TOE, not iWARP). The only valid OP value with this bit is "Send". The total length of the message must be less than or equal to MSS or an AE (AE_WQE_LSMM_TOO_LONG) is generated and the QP state is transitioned to terminate. This bit does not apply to RoCEv2.

wait_for_rcvFPDU (1 bit)

Only valid if *Streaming_Mode* is set to 1b. When this bit is set, the E810 does not process the WQE following this one until a valid inbound iWARP ULPDU is received. This is useful for the last WQE of the IETF response frame. This bit does not apply to RoCEv2.

Inline_Data_Flag (1 bit)

Set to 0b for no inline data.

Remote_Invalidate_STag (32 bits)

The STag on the remote peer that is to be invalidated by the message. This field is only valid if the operation type indicates an invalidate operations.

Tagged_Offset (64 bits), STag (32 bits), Fragment_Length (31 bits), Fragment_Valid (1 bit)

These fields describe the first fragment of the message to be sent. Additional fragments can be added to the WQE by setting *AdditionalFragmentCount* to a non-zero value.

Immed_Data_Flag (1 bit, RoCEv2 only)

When this bit is set (1b), indicates that immediate data is present. The WQE format is modified as follows:

The fragment at offset 0 will contain only the Immediate Data. The SGEs (if any) start in the first additional fragment.

Note: Immediate Data is not allowed for the invalidate operations ("Send with Invalidate" and "Send with Solicited and Invalidate"). For the invalidate operations, the *Immed_Data_Flag* is reserved and should be set to zero by software.

Table 11-79. Immediate Data Format (First Fragment at Offset 0)

Byte Offset	[Bit Range] Field Name			
0	[63:32] RSVD [31:24] Immediate_Data Byte[3] [23:16] Immediate_Data Byte[2]		[15:8] Immediate_Data Byte[1] [7:0] Immediate_Data Byte[0]	
8	[63:0] RSVD			

Immediate_Data (4 bytes, RoCEv2 only)

Byte 0 is the least significant byte of the data and byte 3 is the most significant. The data is specified in little endian format.

11.6.6.1.4 SQ WQE Format - Send with Inline Data

Table 11-80. RDMA SQ Send with Inline Data WQE Format

Byte Offset	[Bit Range] Field Name			
0	[63:0]	Immediate_Data or RSVD		
8	[63:0]	Data		
16	[63:0]	RSVD		
24	[63] [62] [61] [60] [59:58] [57] [56]	WQE_Valid Signaled_Completion Local_Fence Read_Fence RSVD Inline_Data_Flag Push_WQE	[55:48] [47] [46] [45:38] [37:32] [31:0]	Inline_Data_Length Immed_Data_Flag ReportRTT RSVD OP Remote_Invalidate_STag

For the following field definitions, see Section 11.6.6.1.1.

- WQE_Valid (1 bit)
- Signaled_Completion (1 bit)
- Local_Fence (1 bit)
- Read_Fence (1 bit)
- ReportRTT (1 bit)
- Push_WQE (1 bit)

OP (6 bits)

This WQE format is valid for the "Send", "Send with Invalidate", "Send with Solicited Event", and "Send with Solicited Event and Invalidate", as well as the versions of these operations that include inline data. See Section 11.6.6.1.1 for the associated opcode values.

Inline_Data_Flag (1 bit)

The *Inline_Data_Flag* must be set to 1b. This bit specifies that the data is contained inline with the WQE.

Value	Description
0b	The WQE is using descriptors and the WQE format is listed in Table 11-78.
1b	The WQE has inline data and the WQE format is listed in Table 11-80.

Inline_Data_Length (8 bits)

Inline_Data_Length indicates the number of bytes included in the WQE and all subsequent additional fragment descriptors.

Remote_Invalidate_STag (32 bits)

See Section Section 11.6.6.1.3.

Inline_Data

See Section 11.6.6.1.1.2.

Immed_Data_Flag (1 bit, RoCEv2 only)

The *Immed_Data_Flag* set to 1b indicates that immediate data is present.

Note: Immediate Data is not allowed for the invalidate operations ("Send with Invalidate" and "Send with Solicited and Invalidate"). For the invalidate operations, the *Immed_Data_Flag* is reserved and should be set to zero by software.

Immediate_Data (4 bytes)

See Section 11.6.6.1.3.

11.6.6.1.5 SQ WQE Format - RDMA Write

Table 11-81. RDMA SQ RDMA Write WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	Tagged_Offset		
8	[63] [62:32]	Fragment_Valid Fragment_Length	[31:0]	STag
16	[63:0]	Remote_Taged_Offset		
24	[63] [62] [61] [60] [59:58] [57] [56]	WQE_Valid Signaled_Completion Local_Fence Read_Fence RSVD Inline_Data_Flag Push_WQE	[55:48] [47] [46] [45:42] [41:38] [37:32] [31:0]	RSVD Immed_Data_Flag ReportRTT RSVD AdditionalFragmentCount OP Remote_STag

For the following field definitions, see Section 11.6.6.1.1.

- WQE_Valid (1 bit)
- Signaled_Completion (1 bit)
- Local_Fence (1 bit)
- Read_Fence (1 bit)
- Push_WQE (1 bit)
- ReportRTT (1 bit)
- AdditionalFragmentCount (4 bits)

OP (6 bits)

This WQE format is valid for the "Write", "Write with Solicited" opcodes. Write can be used with or without immediate data. Write with Solicited only makes sense if the *Immed_Data_Flag* is on (1b). If the *Immed_Data_Flag* is off (0b), it is the same as a Write. See Section 11.6.6.1.1 for the associated opcode values.

Tagged_Offset (64 bits), STag (32 bits), Fragment_Length (31 bits), Fragment_Valid (1 bit)

These fields describe the first fragment (data source) of the message to be sent. Additional fragments can be added to the WQE by setting *AdditionalFragmentCount* to a non-zero value.

Remote_Tagged_Offset (64 bits), Remote_STag (32 bits)

These fields describe the remote buffer (data sink).

Inline_Data_Flag (1 bit)

Set to 0b for no inline data.

Immed_Data_Flag (1 bit)

The *Immed_Data_Flag* set to 1b indicates that immediate data is present. The WQE format is modified as follows:

The fragment at offset 0 will contain only the Immediate Data. The SGEs (if any) start in the first additional fragment.

Table 11-82. Immediate Data Format (First Fragment at Offset 0)

Byte Offset	[Bit Range] Field Name			
0	[63:56] Immediate_Data Byte[7] [55:48] Immediate_Data Byte[6] [47:40] Immediate_Data Byte[5] [39:32] Immediate_Data Byte[4]		[31:24] Immediate_Data Byte[3] [23:16] Immediate_Data Byte[2] [15:8] Immediate_Data Byte[1] [7:0] Immediate_Data Byte[0]	
8	[63:0] RSVD			

Immediate_Data (8 bytes, iWARP only)

Byte 0 is the least significant byte of the data and byte 7 is the most significant. The data is specified in little endian format.)

Immediate_Data (4 bytes, RoCEv2 only)

Byte 0 is the least significant byte of the data and byte 3 is the most significant. The data is specified in little endian format. Bytes 4-7 are ignored.

11.6.6.1.6 SQ WQE Format - RDMA Write with Inline Data

Table 11-83. RDMA SQ RDMA Write with Inline Data WQE Format

Byte Offset	[Bit Range] Field Name			
0	[63:0] Immed_Data or RSVD			
8	[63:0] Data			
16	[63:0] Remote_Tagged_Offset			
24	[63] WQE_Valid [62] Signaled_Completion [61] Local_Fence [60] Read_Fence [59:58] RSVD [57] Inline_Data_Flag [56] Push_WQE		[55:48] Inline_Data_Length [47] Immed_Data_Flag [46] ReportRTT [45:38] RSVD [37:32] OP [31:0] Remote_STag	

For the following field definitions, see Section 11.6.6.1.1.

- WQE_Valid (1 bit)
- Signaled_Completion (1 bit)
- Local_Fence (1 bit)
- Read_Fence (1 bit)
- ReportRTT (1 bit)
- Push_WQE (1 bit)

OP (6 bits)

See Section 11.6.6.1.1 for the associated opcode value.

Inline_Data_Flag (1 bit)

The *Inline_Data_Flag* must be set to 1b for inline data operations. This bit specifies that the data is contained inline with the WQE.

Value	Description
0b	The WQE is using descriptors and the WQE format is listed in Table 11-81.
1b	The WQE has inline data and the WQE format is listed in Table 11-83.

Inline_Data_Length (8 bits)

Inline_Data_Length indicates the number of bytes included in the WQE and all subsequent additional fragment descriptors.

Remote_Tagged_Offset (64 bits), Remote_STag (32 bits)

These fields describe the remote buffer (data sink).

Inline_Data

See Section 11.6.6.1.1.2.

Immed_Data_Flag (1 bit)

The *Immed_Data_Flag* set to 1b indicates that immediate data is present.

Immed_Data (8 bytes or 4 Bytes)

See Section 11.6.6.1.5.

11.6.6.1.7 SQ WQE Format - RDMA Read

RDMA Read now supports multiple SGEs for iWARP as well as RoCEv2. Read with local invalidate is expanded as well. If a Read with local invalidate is issued with multiple SGEs, only the first STag is invalidated.

Table 11-84. RDMA RDMA Read WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	Tagged_Offset		
8	[63] [62:32]	Fragment_Valid Fragment_Length	[31:0]	STag
16	[63:0]	Remote_Tagged_Offset		
24	[63] [62] [61] [60] [59:58] [57] [56]	WQE_Valid Signaled_Completion Local_Fence Read_Fence RSVD RSVD (Inline_Data_Flag, must be 0) Push_WQE	[55:48] [47] [46] [45:42] [41:38] [37:32] [31:0]	RSVD RSVD (Immed_Data_Flag) ReportRTT RSVD AdditionalFragmentCount OP Remote_STag

For the following field definitions, see Section 11.6.6.1.1.

- WQE_Valid (1 bit)
- Signaled_Completion (1 bit)
- Local_Fence (1 bit)
- Read_Fence (1 bit)
- Push_WQE (1 bit)
- ReportRTT (1 bit)

OP (6 bits)

This WQE format is valid for the RDMA read and RDMA read with local invalidate operations. See Section 11.6.6.1.1 for the associated opcode value.

Tagged_Offset (64 bits), STag (32 bits), Fragment_Length (31 bits), Fragment_Valid (1 bit)

These fields describe the local target (data sink) of the RDMA read operation. STag also specifies the local STag to be invalidated for RDMA read with local invalidate operations.

Remote_Tagged_Offset (64 bits), Remote_STag (32 bits)

These fields describe the remote buffer (data source).

If the *infiniband_read_en* bit is set in the iWARP context, this QP supports new semantics for RDMA Read. See Section 11.4.1.5.2.2.1 for a description of the capabilities.

AdditionalFragmentCount (4 bits)

See Section 11.6.6.1.1.

The *AdditionalFragmentCount* field is reserved for iWARP QPs if *infiniband_read_en* is off (0b).

11.6.6.1.8 SQ WQE Format - Memory Window Bind

Table 11-85. SQ Bind Memory Window WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	Memory_Window_Base_VA		
8	[63:32]	Parent_Memory_Region_STag	[31:0]	Memory_Window_STag
16	[63:46]	RSVD	[45:0]	Memory_Window_Length
24	[63] [62] [61] [60] [59:57] [56] [55]	WQE_Valid Signaled_Completion Local_Fence Read_Fence RSVD Push_WQE RSVD	[54] [53] [52:48] [47] [46:38] [37:32] [31:0]	Memory_Window_Type VA_Based_TO Access_Rights RSVD (Immed_Data_Flag) RSVD OP RSVD

For the following field definitions, see Section 11.6.6.1.1.

- WQE_Valid (1 bit)
- Signaled_Completion (1 bit)
- Local_Fence (1 bit)
- Read_Fence (1 bit)
- Push_WQE (1 bit)

OP (6 bits)

See Section 11.6.6.1.1 for the associated opcode value.

Memory_Window_Base_VA (64 bits)

This field specifies the starting point of the memory window within the parent memory region's virtual address range for the memory window.

Memory_Window_Length (46 bits)

This field specifies the size of the memory window. Specifying 0b as a length results in an AE code of AE_AMP_MWBIND_INVALID_BOUNDS for Type 2 Windows. Specifying 0b as a length results in an invalidate of a Type 1 Window.

Memory_Window_STag (32 bits)

This field specifies STag of the memory window including the index and additional keys.

If the Window is a Type 1 Window, the key value (STag/R_Key) must change. Software will change the random bits and/or the user key, but the index field must not change. Hardware uses the index to locate the MRTE and will change the key value. Since the index field does not change, the same MRTE continues to be used.

Memory_Window_Type (1 bit)

Value	Description
0b	The entry describes a Type 2B memory window.
1b	The entry describes a type 1 memory window.

Parent_Memory_Region STag (32 bits)

This field specifies STag of the parent memory region to which this memory window is bound including the index and additional keys.

VA_Based_TO (1 bit)

VA_Based_TO specifies if the memory window is zero-based or VA-based.

Value	Description
0b	The memory window is zero-based
1b	The memory window is VA-based

Access_Rights (5 bits)

Indicates the rights assigned to this STag. The valid bits for this field are:

Value	Description
00100b	Enable remote read.
01000b	Enable remote write
Note: All other values are reserved.	

11.6.6.1.9 SQ WQE Format - Local Invalidate

Table 11-86. Local Invalidate WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	RSVD		
8	[63:32]	RSVD	[31:0]	STag
16	[63:0]	RSVD		
24	[63] [62] [61] [60] [59:57] [56]	WQE_Valid Signaled_Completion Local_Fence Read_Fence RSVD Push_WQE	[55:48] [47] [46:38] [37:32] [31:0]	RSVD RSVD (Immed_Data_Flag) RSVD OP RSVD

For the following field definitions, see Section 11.6.6.1.1.

- WQE_Valid (1 bit)
- Signaled_Completion (1 bit)
- Local_Fence (1 bit)
- Read_Fence (1 bit)
- Push_WQE (1 bit)

OP (6 bits)

See Section 11.6.6.1.1 for the associated opcode value.

STag (32 bits)

STag specifies the STag to be invalidated.

11.6.6.1.10 SQ WQE Format - Fast Register

Fast register support for the E810 has two modes of operation that depend on the number of available PBL resources. The prerequisite for issuing a fast register operation is to allocate an invalid memory region with a specific number of page list entries (or PBLEs for the E810). Since the E810 allows host software to directly populate PBLs for memory registration, it is desirable to keep the same approach for fast register operations. The issue with allowing only the mode where host software directly populates the PBLs for fast register is apparent if an application issues fast register, send, local invalidate, fast register operations using the same memory region without waiting for completions for the local invalidate operation. If the E810 only allowed the mode where software directly populated PBLs for fast register operations in these cases, memory corruption would be highly likely to occur. Software mechanisms to resolve the races end up with extremely inefficient fast register operations if they could be made to work at all. To not penalize every application for the previous behaviors, but still provide proper handling of these behaviors, the following algorithm should be used for fast register operations:

1. During the allocate memory region operation, software must reserve the full number of PBL resources necessary to satisfy the allocate memory region request.
2. Software must keep a reference count per allocated memory region to track the number of outstanding fast register operations that are outstanding against the memory region that used the PBL resources allocated during the allocate memory region operation.
3. On fast register operation, if there are enough free PBL resources to satisfy the new request, software should populate a new area of PBLEs and issue the fast register operation without setting the *Copy_Host_PBLs* flag.
4. If there are not enough free PBL resources to satisfy the new request but the reference count of the users of the PBLs allocated during the allocate memory region operation is zero, software can use the PBL resources but must increment the reference count for the memory region.
5. If there are not enough free PBL resources to satisfy the new request but the reference count is one or more, host software must allocate a pinned buffer large enough to hold the page list specified on the fast register operation, populate that buffer with the page list specified by the fast register operation, and set the *Copy_Host_PBLs* bit and increment the reference count.
6. When the memory region associated with the fast register has been invalidated, the reference count for the memory region must be decremented.

This algorithm minimizes that cases where the page list associated with a fast register operation must be copied across the PCI bus while maintaining safe access to the memory region state and page list.

Table 11-87. SQ Fast Register WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	Virtual_Address_or_First_Buffer_Offset		
8	[63:12]	PBL_Address	[11:0]	first_pm_pbl_index_high
16	[63:48] [47:46]	first_pm_pbl_index_low RSVD	[45:0]	STag_Length
24	[63] [62] [61] [60] [59:57] [56] [55:54] [53]	WQE_Valid Signaled_Completion Local_Fence Read_Fence RSVD Push_WQE RSVD VA_Based_TO	[52:48] [47:46] [45:44] [43] [42:38] [37:32] [31:8] [7:0]	Access_Rights Host_Page_Size Leaf_PBL_Size Copy_Host_PBLs RSVD OP Driver_Key_STag_Index Consumer_Key

For the following field definitions, see Section 11.6.6.1.1.

- WQE_Valid (1 bit)
- Signaled_Completion (1 bit)
- Local_Fence (1 bit)
- Read_Fence (1 bit)
- Push_WQE (1 bit)

OP (6 bits)

See Section 11.6.6.1.1 for the associated opcode value.

VA_Based_TO (1 bit)

VA_Based_TO specifies if the STag is zero-based or VA-based.

Value	Description
0b	The memory window is zero-based.
1b	The memory window is VA-based.

Zero-based STags carry only the first buffer offset in the *Virtual_Address_or_First_Buffer_Offset* field. VA-based STags carry the full base VA including first buffer offset in the *Virtual_Address_or_First_Buffer_Offset* field.

Virtual_Address_or_First_Buffer_Offset (64 bits)

Indicates the base VA for this region/window for VA-based entries and indicates the first buffer offset for zero-based entries.

STag_Length (46 bits)

Length of the memory region or memory windows specified by the STag index specified by *Driver_Key_Stag_Index*. Specifying 0 as a length results in an AE code of AE_AMP_FASTREG_INVALID_LENGTH. If *Leaf_PBL_Size* is set to 1b and *Host_Page_Size* = 0, AE_AMP_FASTREG_INVALD_LENGTH is also generated if the size exceeds 228 x 4096, or 2^7 host page sizes for VMs.

Access_Rights (5 bits)

Indicates the rights assigned to this STag. The values for this field are:

Value	Description
00001b	Enable local read.
00010b	Enable local write.
00100b	Enable remote read.
01000b	Enable remote write.
10000b	Enable window bind.
Note: All other values are reserved.	

Host_Page_Size (2 bits)

Host_Page_Size specifies the page size of the backing pages for the STag. The values for this field are:

Value	Description
00b	4 KB pages.
01b	2 MB pages.
10b	1 GB pages.
11b	Reserved.

Driver_Key_STag_Index (24 bits)

Index and *Driver Key* fields of the STag associated with the memory region. The E810 supports a variable size STag index. This means that the number of bits used for *Driver Key* and for *STag_Index* are dependent on the maximum number of STags supported for a given PCI function. For example, if a PCI function read the FPMPEMRSZ field (See Section 13.2.2.20.130) and found that the maximum number of MRTEs was 64 KB, the lower 16 bits of this field would be the STag index and the upper 8 bits would be a driver key that the driver can randomize to make guessing the MRTE layout more difficult to guess.

Consumer_Key (8 bits)

Consumer_Key is the least significant 8-bit portion of the STag. This field is supplied by the user application or the driver.

Copy_Host_PBLs (1 bit)

PBLs for an STag are located in host memory for the E810 in the pages allocated for the E810 HMC. In most situations, software copies backing pages for an STag directly to the HMC pages to optimize performance. If software needs the E810 to populate the HMC pages, *Copy_Host_PBLs* must be set to 1b and *PBL_Address* must point to the physical address of the backing pages in host memory.

Note: All PBLEs (root and leaf) must be allocated in chunks of eight.

Leaf_PBL_Size (2 bits)

The E810 supports physically-contiguous STags and two forms of virtually-contiguous STags. Physically-contiguous STags do not require any PBLs and store physical address of the first page of the STag directly with the STag (no leaf PBL). Virtually-contiguous STags that can be represented with a single HMC virtually contiguous address range require a single-level PBL of Variable size. Virtually-contiguous STags that are large (or in cases where the HMC address space for PBLs becomes fragmented) might require two level PBLs. In this case, the E810 needs to know the length of the leaf PBLs to properly manage access to the PBLs. The valid settings for *Leaf_PBL_Size* are the following:

Value	Description
00b	No leaf PBL.
01b	Variable (one level).
10b	256 bytes (two level).
11b	4 KB (two level).

PBL_Address (52 bits)

Only valid if *Copy_Host_PBLs* is set (1b) or if *Leaf_PBL_Size* is set to 0b. If *Copy_Host_PBLs* is set and *Leaf_PBL_Size* is not set to 0b, PBL address is the physical address of the PBLs in host memory if they were not copied to HMC pages by software. If *Leaf_PBL_Size* is 256 or 4 KB, PBL address contains the physical address of a packed array of Root PBLEs. The format of the Root PBLEs is listed in Table 11-88.

If *Leaf_PBL_Size* is variable, *PBL_Address* contains the physical address of the page list in host memory to be copied to *first_pm_pbl_index*. If *Copy_Host_PBLs* is clear and *Leaf_PBL_Size* is set to 0b, the physical address of the physically contiguous memory is contained in this field.

Table 11-88. Format of Root PBLEs in Host Memory for Two-Level PBLs

Byte Offset	[Bit Range] Field Name			
0	[63:0]	Leaf_PBL_Address		
8	[63:28]	RSVD	[27:0]	first_pm_pbl_index

first_pm_pbl_index (28 bits)

This field defines the HMC PBLE object index used for the memory region page list. If *Copy_Host_PBLs* is set (1b), the E810 copies the page list from the host address specified by *PBL_Address* to the HMC object specified by *first_pm_pbl_index*. If *Copy_Host_PBLs* is clear (0b), *first_pm_pbl_index* designates the HMC base address for the PBLs for this STag that software has already initialized with PBL information.

11.6.6.1.11 SQ WQE Format - Connection Established

Table 11-89. SQ Fast Register WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	RSVD		
8	[63:0]	RSVD		
16	[63:0]	RSVD		
24	[63] [62] [61:38]	WQE_Valid RSVD (Signaled_Completion) RSVD	[37:32] [31:0]	OP RSVD

For the following field definitions, see Section 11.6.6.1.1.

- WQE_Valid (1 bit)
- Signaled_Completion (1 bit)

OP (6 bits)

This WQE tells the device to notify software when the connection is established. An AE (AE_LLP_CONNECTION_ESTABLISHED) informs software that the connection is complete. This WQE is not valid for RoCEv2 or UDA QPs. See Section 11.6.6.1.1 for the associated opcode value.

11.6.6.2 RQ WQE Format

Table 11-90. iWARP RQ WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	Tagged_Offset		
8	[63] [62:32]	Fragment_Valid Fragment_Length	[31:0]	STag
16	[63:0]	RSVD		
24	[63] [62:42] [41:38]	WQE_Valid RSVD AdditionaFragmentCount	[37:32] [31:0]	RSVD (OP) RSVD

WQE_Valid (1 bit) and AdditionalFragmentCount (4 bits)

See Section 11.6.6.1.1.

Tagged_Offset (64 bits), STag (32 bits), Fragment_Length (31 bits), Fragment_Valid (1 bit)

These fields describe the first fragment of the byte stream to be sent. Additional fragments can be added to the WQE by setting AdditionalFragmentCount to a non-zero value.

11.7 UD/UDA Functionality

The operations for RoCEv2 UD and UDA are implemented using the WQE formats described in the following sections. QP operation, the verbs interface, and system view of the E810 are described in Section 11.4.1. The CQP operations and HMC structures necessary to bring the E810 to a functional state for RDMA are described in Section 11.5.

11.7.1 UD/UDA Descriptor Formats

11.7.1.1 UD/UDA CQ Entry Formats

CQ operation for RDMA is described in Section 11.4.1.3. CQs are manipulated through CQP operations. See Section 11.5.3.3 for further details.

For RQ completions, see Section 11.7.1.4 for a description of the received data.

When *Avoid_Memory_Conflicts* is not set (0b), the CQE is normally 32 bytes. However, some completions require additional information. In this case, the *Extended_CQE* bit is set (1b) and an additional 32 bytes is added just for that CQE.

When *Avoid_Memory_Conflicts* is set (1b), the CQE is always 64 bytes. If the *Extended_CQE* bit is not set (0b), the second half of the CQE is all zeros. If the *Extended_CQE* bit is set (1b), the second half of the CQE is described in Table 11-92.

Table 11-91. CQ Entry Format from UD/UDA QP

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:32]	Packet_Sequence_Number_or_RTT	[31:0]	Payload_Length
8	[63:0]	QP_Completion_Context		
16	[63:50] [49:32]	RSVD QP_ID	[31:24] [23:0]	RSVD UD_Src_QPN
24	[63] [62] [61:56] [55] [54] [53] [52]	CQE_Valid SQ OP Error Solicited_Event (RQ Only) IPv4 RSVD	[51] [50] [49:47] [46:32] [31:16] [15:0]	Push_Dropped (SQ Only) Extended_CQE RSVD WQ_Desc_Index Major_Error_Code Minor_Error_Code

Table 11-92. Extended CQ Entry Format from UD QP

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
32	[63:32] [31:24] [23:16]	RSVD Immediate_Data Byte[3] Immediate_Data Byte[2]	[15:8] [7:0]	Immediate_Data Byte[1] Immediate_Data Byte[0]
40	[63:0]	RSVD		
48	[63:48]	UD_VLAN_Tag	[47:0]	UD_smac
56	[63] [62] [61]	CQE_Valid2 Immediate_Data_valid UD_smac_valid	[60] [59:0]	UD_VLAN_Tag_valid RSVD

OP (6 bits)

This field reports the opcode from the operation associated with the CQE.

Extended_CQE (1 bit)

When the *Extended_CQE* bit is set (1b), the CQE is extended to 64 bytes. The second half of the CQE format is shown in Table 11-92. When the Extended_CQE is valid, software must ensure that the *CQE_Valid2* flag has the right polarity. If it is not valid, the CQE is not complete and software cannot yet process the CQE.

WQ_Desc_Index (15 bits)

WQ is sliced up into 32-byte descriptor quanta. Every WQE must start with a 32-byte descriptor on a 32-byte boundary. *WQ_Desc_Index* reports the 32-byte quanta index of the WQE associated with the completion.

CQE_Valid (1 bit)

The *CQE_Valid* bit for CQE indicates that a CQE is ready to be processed. The polarity of the *Valid* bit changes each time the CQ wraps from the last entry back to the first entry. This change in polarity reduces software overhead by avoiding the need to clear the *Valid* bit once software has processed a valid CQE. Software is responsible to clear (set to 0b) all memory in a CQ initially at CQ creation. The first iteration (and subsequent odd numbered iterations) through the CQ, the E810 sets the *Valid* bit to 1b when it writes a new CQE. For the second iteration (and all even numbered iterations) through the CQ, the E810 sets the *Valid* bit to 0b when it writes an new CQE.

Error (1 bit)

Value	Description
0b	No error.
1b	An error occurred when processing the WQE associated with this CQE and that the *Error_Code* field is valid.

Solicited_Event (1 bit)

Valid for receive only. Value for SQ completions is undefined.

Value	Description
0b	SE bit not set in received packet.
1b	SE bit set in received packet.

IPv4 (1 bit)

This field indicates that the message received was IPv4 or IPv6.

Value	Description
0b	IPv6
1b	IPv4

SQ (1 bit)

Value	Description
0b	RQ
1b	SQ

Push_Dropped (1 bit)

Valid only when SQ is set.

Value	Description
0b	Push operations are being processed successfully.
1b	A recent push mode operation has been dropped by the E810. Software should refrain from submitting additional push mode operations until the SQ has encountered an empty condition.

Payload_Length (32 bits)

Total payload length of the completed message. This field is only valid for RQ WQE completions. The length includes the 40 bytes for the IP header. For UDA QP, the length also includes the L4 header.

Packet_Sequence_Number_or_RTT (32 bits)

For a RoCEv2 SQ completions, this field is the Packet Sequence Number or RTT.

QP_Completion_Context (64 bits)

Completion context pointer. This field is transferred to the CQE from QP context.

QP_ID (18 bits)

QP associated with the completed message.

UD_Src_QPN (24 bits)

QP number of the sender. This is valid only for UD QPs.

Major_Error_Code (16 bits)

Valid if *Error* bit is set (1b). See Table 11-69 for defined values. Software can also report any value via the Flush WQEs CQP operation (see Section 11.5.3.17).

Minor_Error_Code (16 bits)

Valid if *Error* bit is set (1b). See Table 11-69 for defined values. Software can also report any value via the Flush WQEs CQP operation (see Section 11.5.3.17).

The following items are in the extended CQE:

CQE_Valid2 (1 bit)

The *CQE_Valid2* bit must match the setting for a valid CQE in the CQ. Normally the value of the *CQE_Valid2* bit will be the same as the *CQE_Valid* bit in this completion. However, there is one case where it is different: if the completion starts in the last entry of the CQ, the second half of the CQE wraps to the start of the CQ, so the *CQE_Valid2* bit flips to match the *CQE_Valid* setting for the next pass of the CQ.

Immediate_Data_valid (1 bit)

This bit indicates that immediate data was received.

For RoCEv2, the *Immediate_Data* field is four bytes, which are in bytes 0-3 the *Immediate_Data* field and bytes 4-7 are ignored.

UD_smac_valid (1 bit)

Indication that the MAC Address of the sender is valid. This is set only if the *UDPrivCQEnable* flag is set in the QP context.

UD_smac (48 bits)

The MAC Address of the sender. This is only set if *UD_smac_valid* is set.

UD_VLAN_Tag_valid (1 bit)

Indication that the *VLAN_Tag* field is valid. This value is set only if the *UDPrivCQEnable* flag is set in the QP context. Port VLAN must not be reported.

UD_VLAN_Tag (16 bits)

VLAN Tag of the sender. This value is set only if *UD_VLAN_Tag_valid* is set.

11.7.1.2 UD/UDA SQ Descriptors

The following WQE formats are used in conjunction with UD QPs. Operations that are supported for UD QPs are the listed in Table 11-93.

The NOP operation is unchanged (see Section 11.6.6.1.2).

Table 11-93. UD/UDA QP Operations

Operation Code	Operation Name	Section Reference
0x00-0x02	Reserved	N/A
0x03	Send	11.7.1.3.3
0x04	Reserved	N/A
0x05	Send with Solicited Event	11.7.1.3.3

Operation Code	Operation Name	Section Reference
0x06-0x0B	Reserved	N/A
0x0C	NOP	11.7.1.3.2
0x0D-0x3F	Reserved	N/A

11.7.1.3 UD/UDA SQ Descriptor Formats

11.7.1.3.1 Common UD/UDA SQ Descriptor Format Fields

The basic RDMA WQE is a 32-byte structure that is broken up into 64-bit (8-byte) words. The placement of these fields with WQE are common among all CQP WQEs, RDMA WQEs and also UD WQEs. Section 11.5.3.1 lists the basic structure of a CQP WQE, including the common fields.

The definition of fields marked as “Operation Code Dependent” vary from operation to operation and are detailed in subsequent sections. Fields marked as “Reserved” must be set to 0b or undesired behavior related to the specific QP associated with the WQE might occur.

Additionally, RDMA WQEs can optionally include additional fragments to enable larger WQEs to be created.

The only UD operation defined for the SQ is Send (immediate data and inline data are allowed).

11.7.1.3.2 UD/UDA SQ WQE Format - NOP

See Section 11.6.6.1.2.

11.7.1.3.3 UD/UDA SQ WQE Format - Send

For UDA queues, the first fragment must contain the TCP or UDP header. The payload can start in the first fragment after the header or it can begin in the second fragment.

Table 11-94. UD/UDA SQ Send WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	Tagged_Offset		
8	[63] [62:32]	Fragment_Valid Fragment_Length	[31:0]	STag
16	[63:56] [55:32]	RSVD Dest_QPN	[31:0]	Dest_QKey
24	[63] [62] [61] [60:58] [57] [56] [55:48]	WQE_Valid Signaled_Completion UDP_Header RSVD Inline_Data_Flag Push_WQE RSVD	[47] [46] [45:42] [41:38] [37:32] [31:17] [16:0]	Immed_Data_Flag RSVD L4LEN AdditionalFragmentCount OP RSVD AH_ID

For the following field definitions, see Section 11.6.6.1.1.

- WQE_Valid (1 bit)
- Signaled_Completion (1 bit)
- Push_WQE (1 bit)
- AdditionalFragmentCount (4 bits)

OP (6 bits)

This WQE format is valid for the "Send" and "Send with Solicited Event" as well as the versions of these operations that include immediate data. See Section 11.7.1.2 for the associated opcode values.

Send with solicited and immediate data are not valid for UDA QPs.

Inline_Data_Flag (1 bit)

Set to 0b for no inline data.

Tagged_Offset (64 bits), STag (32 bits), Fragment_Length (31 bits), Fragment_Valid (1 bit)

These fields describe the first fragment of the message to be sent. Additional fragments can be added to the WQE by setting *AdditionalFragmentCount* to a non-zero value.

AH_ID (17 bits)

Index of Address Handle as allocated by software. This index together with HMC configuration for the PCI function identifies location of the hardware Address Handle structure in the host memory.

Dest_QPN (24 bits)

QP Number on the remote machine.

Dest_QKey (32 bits)

If the most significant bit is not set, this is an unprivileged Q_Key. The hardware will send this value for the destination Q_Key.

If the most significant bit is set, this is a privileged Q_Key so hardware will not use this value. Instead, hardware will send the Q_Key that is in the QP context.

Immed_Data_Flag (1 bit)

The *Immed_Data_Flag* set to 1b indicates that immediate data is present. The WQE format is modified as follows:

The fragment at offset 0 will contain only the Immediate Data. The SGEs (if any) start in the first additional fragment.

The Immed_Data_Flag is only valid for UD QPs, It is ignored for UDA queues.

UDP_Header (1 bit)

Value	Description
0b	Indicates that request is sent using TCP.
1b	Indicates that request is sent using UDP.

Software must supply the UDP/TCP header at beginning of the first fragment. The payload can start in the first fragment after the header or it can start in the second fragment.

The *UDP_Header* flag is valid only for UDA QPs. It is ignored for UD QPs.

L4LEN (4bits)

The L4 header length in DWords. It should be set to 8/12 for UDP, respectively, and be equal to or larger than 5 (or 20 bytes) for TCP.

The L4LEN field is valid only for UDA QPs. It is ignored for UD QPs.

Table 11-95. Immediate Data Format (First Fragment at Offset 0)

Byte Offset	[Bit Range] Field Name			
0	[63:32] RSVD [31:24] Immediate_Data Byte[3] [23:16] Immediate_Data Byte[2]		[15:8] Immediate_Data Byte[1] [7:0] Immediate_Data Byte[0]	
8	[63:0] RSVD			

Immediate_Data (4 bytes)

Byte 0 is the least significant byte of the data and byte 3 is the most significant. The data is specified in little endian format.

11.7.1.3.4 UD/UDA SQ WQE Format - Send with Inline Data

For UDA queues, the inline data must start with the TCP or UDP header. The payload starts after the header.

Table 11-96. UD/UDA SQ Send with Inline Data WQE Format

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
0	[63:0]	Immediate_Data or RSVD		
8	[63:0]	Data		
16	[63:56] [55:32]	RSVD Dest_QPN	[31:0]	Dest_QKey
24	[63] [62] [61] [60:58] [57] [56] [55:48]	WQE_Valid Signaled_Completion UDP_Header RSVD Inline_Data_Flag Push_WQE Inline_Data_Length	[47] [46] [45:42] [41:38] [37:32] [31:17] [16:0]	Immed_Data_Flag ReportRTT L4LEN RSVD OP RSVD AH_ID

For the following field definitions, see Section 11.6.6.1.1.

- WQE_Valid (1 bit)
- Signaled_Completion (1 bit)
- Push_WQE (1 bit)

For the following field definitions, see Section 11.7.1.3.3.

- Dest_QPN (24 bits)
- Dest_QKey (32 bits)
- UDP_Header (1 bit)
- Immediate_Data (4 Bytes)
- Immed_Data_Flag (1 bit)
- L4LEN (4 bits)
- AH_ID (16 bits)
-

OP (6 bits)

This WQE format is valid for the send and send with solicited event. See Section 11.7.1.2 for the associated opcode values.

Send with solicited and immediate data are not valid for UDA QPs.

Inline_Data_Flag (1 bit)

The *Inline_Data_Flag* must be set to 1b. This bit specifies that the data is contained inline with the WQE.

Value	Description
0b	The WQE is using descriptors and the WQE format is listed in Table 11-78.
1b	The WQE has inline data and the WQE format is listed in Table 11-80.

Inline_Data_Length (8 bits)

Inline_Data_Length indicates the number of bytes included in the WQE and all subsequent additional fragment descriptors.

Inline_Data

See Section 11.6.6.1.1.2.

11.7.1.4 UD/UDA RQ Descriptors

The RQ descriptors are not changed. See Section 11.6.6.2 for details.

When a receive completion arrives, the first 40 bytes of the receive buffer contains the IP header of the incoming packet. If the packet was sent using IPv6, the IP header consumes the entire 40 bytes. If the packet was sent using IPv4, the IP header is placed in the second 20 bytes. That is, the IP header consumes bytes 20 through 39 of the receive buffer. The contents of the first 20 bytes is undefined.

For UDA only, in addition to the 40-byte IP header area, the L4 header is included in the receive buffer.

The form of the UD/UDA received buffer is:

- 40-byte area for the IP header (same for UD and UDA).
- UDA QP receive buffers have the L4 header (e.g. TCP or UDP header). This is not present for UD QP receive buffers.
- Payload

11.8 UDA Functionality

Userspace Direct Access (UDA) was intended to provide userspace access queues in a general way, but this feature is not supported in the E810. UDA is available only in the kernel and is limited to iWARP connection setup and error handling. UDA is not available in user space.

The UDA host interface and semantics are very similar to one exposed by RDMA.

- Sending and receiving data.
- Application uses UDA QPs and CQs to send and receive data, see Section 11.5.3.2 and Section 11.5.3.3 for UDA QP and CQ management requests. UDA QPs and CQs resources are shared with RDMA.
- Completion and AE management.
- CE notification mechanism is identical to one used for RDMA, see Section 11.5.3.11 and Section 11.5.3.12 for CEQ and AEQ allocation description. UDA can share event and completion notification structures with RDMA or have its own resources.
- Application buffer memory management.
- An application should register its buffers (see Section 11.5.3.4) to enable direct placement to and from application buffers. Memory registration used for UDA buffers is identical to the memory registration of RDMA, and uses resources from the same pool of memory regions available for PCI function.
- UDA enables send and receive Ethernet frames by posting frame payload and ULP protocol headers to UDA QP using post send and post receive operations described in Section 11.8.8.1 and Section 11.8.8.4. Separate SQ/RQ WQE is used to send and receive single Ethernet frame.

UDA provides a protocol agnostic interface with limited stateless accelerations for the well known protocols. To enable secure userspace networking for non-privileged consumers UDA limits scope of supported protocols to IP-based protocol only, by offloading Ethernet and IP header generation to hardware. Address Handles (see Section 11.5.3.13) are used to provide hardware with information required to generate Ethernet and IP headers.

Hardware provides partial stateless acceleration for the limited scope of protocols, discussed in Section 11.8.1 and Section 11.8.2 later.

UDA acceleration enables a wide scope of applications to take advantage of direct communication with hardware bypassing system overhead. UDA brings a variety of benefits, including deterministic and low latency, low-latency deviation and jitter, high-message processing rate with a very low network communication overhead, lock-free networking, and linear scalability with number of cores in the system without compromising system security.

11.8.1 Transmit UDA Hardware Acceleration

Transmit UDA hardware provides a limited number of partial stateless accelerations.

UDA traffic is not expected to use a dedicated internal switching resources (VSIs). It shares an internal switch virtual port with RDMA traffic. Depending on the assignment of the user priority, UDA traffic might share a QS with RDMA QPs, or use a dedicated QS. Assignment of UDA traffic to the QS is done at QP creation time, and is similar to assignment of RDMA QP.

The E810 generates Ethernet and IP headers based on information provided in address handle for IP-based userspace protocols (see Section 11.5.3.2). Offloading Ethernet and IP headers generation does not bring much performance improvement, but does improve security of UDA for IP-based protocols. In the standard deployment configuration, UDA supports only IP-based protocols, with an option available for the privileged software running in trusted environment allowing to generate all headers by software. Privileged mode can be enabled by setting *Privileged Header Generation Enable* bit in the UDA QP context, see Section 11.8.4.

The E810 supports header generation for non-fragmented datagrams. Hardware is responsible for the *IP Identification* field. Hardware maintains one instance of the *IP Identification* field per UDA QP.

Software is responsible for generating upper protocol headers (UDP/TCP, and so on). Those headers can be provided as a part of the payload WQE fragment or as a separate WQE fragment. In privileged mode, software can provide headers in the same WQE fragment with payload or using one or more dedicated WQE fragments. The E810 also supports an option of inline data, when both upper layer protocol headers and a frame payload are copied directly to the WQE (see Section 11.8.8.2 for more details). Push mode, described in Section 11.4.1.5.6, can be used to transmit UDA WQEs with inline data (see Section 11.8.8.2 for more details).

The E810 offloads checksum generation for non-fragmented UDP and TCP packets.

UDA traffic uses standard internal switching capabilities provided by the E810 to internally switch traffic between different VSIs. Unlike traffic generated by standard operating system stack, UDA QPs also require additional internal switching capabilities within VSI, similar to RDMA. This enables internal switching between userspace processes directly communicating with hardware using UDA QPs. This capability is enabled by setting a *DoLoopback* bit in the Address Handle corresponding to the destination residing on the same VSI. Physical Function device driver is responsible for allocation and management of Address Handles, also responsible for setting and clearing this bit in previously-allocated Address Handles. For the unicast traffic this bit is set to 1b, at Address Handle allocation time, and indicates that all UDA packets generated using this Address Handle are internally switched.

UDA and the host operating system networking stack independently generates the *IP Identification* field. To avoid collision of the IP identification space, the E810 has a configurable option to override *IP Identification* fields generated by the host networking stack and UDA and forces the MSB of the IP identification to different values. The *IP Identification* field override can be enabled per function using IPCONFIG.*USEENTIREIDRANGE* register/bit.

11.8.2 Receive UDA Hardware Filtering and Acceleration

Receive UDA acceleration is focusing on identifying UDA frames and delivering those frames to the associated UDA RQ. This section describes how UDA frames can be identified using various E810 filters, lists E810 filters that can be programmed for UDA traffic identification, and refers to the sections describing programming of each filter in details. UDA E810 filtering descriptions are organized by the traffic types.

11.8.2.1 UDP UDA Filtering and Acceleration

The E810 supports UDA acceleration of the UDP non-fragmented datagrams. Each VSI has a configuration bit that allows to enable UDA UDP acceleration for that VSI. Once enabled, UDP packets forwarded by the E810 internal switching fabric to the VSI are forwarded for the further filtering and processing by PE.

Figure 11-2 on page 1376 shows filtering steps that apply to all traffic types accelerated by Protocol Engine. Following is a description of each one of those steps applied to UDA UDP traffic.

Each VSI allows enabling of UDA UDP acceleration. When enabled, all UDP packets forwarded to such VSI are forwarded for the further processing by PE Filters associated with VSI. If UDA UDP is disabled, all packets forwarded to VSI are processed by LAN. The E810 uses internal switching forwarding tables to forward packets to VSIs.

UDP traffic is processed by UDA filters associated with PCI function that owns VSI (either VF or PF). If PE filters do not accept packet, the packet is forwarded to the LAN logic for further processing.

The Accelerated Port Table (see Section 11.5.3.18) is used by UDA in conjunction with other accelerated traffic types (iWARP) to filter inbound traffic based on the transport protocol destination port. UDA uses this table for UDP and TCP protocol filtering. When UDA QP expects to receive accelerated datagrams on the particular port (result of bind local port operation), UDA software should request to update the Accelerated Port Table (see Section 11.5.3.18). Since the Accelerated Port Table is shared by multiple PCI functions, false positive identification is possible. The Accelerated Port Table applies to filtering non-fragmented UDP datagrams only, or first fragments. Non-first fragments are not forwarded to the Protocol Engine.

If a UDP datagram is allowed by the Accelerated Port Table, it is filtered through perfect Hash Table (see Section 11.5.3.20). When UDA software binds to the local port, it should add a new entry to the Hash table belonging to the respective PCI function. Binding to the local port, software should add an entry to the Hash Table belonging to VF or PF respectively. Actual CQP request to update the Hash Table (see Section 11.5.3.20) should be done by PF driver only. This operation assumes communication between VF and PF UDA drivers.

The Hash Table is used to filter UDP datagrams belonging to the accelerated UDA QPs, and resolve QP for the UDP datagrams. The Hash filter uses different packet header fields to match Quad Hash table entries depending on the protocol type:

- Unfragmented UDP unicast datagrams are matched based on the destination MAC Address, innermost VLAN tag (if present), destination IP Address, and destination UDP port.
- Fragmented UDP packets are not forwarded to the Protocol Engine.

A packet forwarded to UDA QPs are placed to the application buffers provided in RQ WQEs. UDA places the entire Ethernet frame to the buffers provided by the application, except for L2 Tags that can be stripped depending on VSI configuration. See Section 11.8.8.4 for description of RQ WQE format.

A single Ethernet frame consumes one RQ WQE. RQ WQE should have enough buffering for the header and packet payload. The most inner VLAN Tag is provided in completion along with other information describing received packet. See Section 11.8.6.

If the size of the packet header or payload exceeds buffers provided by software, UDA RQ can be configured to:

- Truncate the packet, or the header, and report an error in corresponding Completion Queue Entry (CQE).
- Silently drop the packet, and do not consume the WQE even if packet/header placement is started. Effectively hardware should pretend that packet was never received and unroll the pointers.

If UDA RQ is empty (i.e., software is lagging behind and did not post RQ WQE in time), the inbound packet is dropped, and the GLPES_PFIP4RXDISCARD statistics counter is incremented. E810 RQs can be configured to a large number of WQEs that can be pre-posted by software, in addition to the internal chip buffers that can be flexibly configured to provide additional buffering space. These two factors should minimize chances of intermittent failures by software to pre-post receive descriptors in time.

In addition to filtering inbound accelerated UDP traffic to one or more UDA QPs, the E810 provides several hardware accelerations, described below.

- **Checksum validation for the non-fragmented UDP datagrams** — The E810 calculates and validates IP and UDP checksums for the non-fragmented UDP/IP datagrams. If the checksum does not match, the Ethernet frame is dropped and not delivered to UDA RQ.

11.8.2.2 TCP UDA Filtering and Acceleration

TCP UDA acceleration enables taking advantage of UDA QPs to send and receive unfragmented TCP segments directly from the application address space or from the kernel. iWARP connection management and exception handling is the usage model for this capability.

On the transmit side, TCP UDA acceleration can take advantage of the generic UDA acceleration capabilities described in Section 11.8.1.

This section describes receive filtering capabilities and hardware accelerations that can be used to accelerate TCP segments.

UDA TCP acceleration is enabled as soon as iWARP capabilities are enabled for the E810. Similar to iWARP, PCI function for the inbound TCP segments are identified using destination MAC Address.

UDA TCP acceleration does not support fragmented IP datagrams. All fragments are processed by stateless filters and delivered to the standard host stack.

UDA TCP acceleration uses an accelerated port table to filter out accelerated traffic based on the destination TCP port. Software should allocate a port to be used for UDA TCP accelerated traffic using the CQP operation described in Section 11.5.3.2.

TCP packets that hits an accelerated port table are filtered through the quad hash table.

The quad hash table is a perfect filter that carries an entry for each accelerated TCP connection that is offloaded to the E810's PE. The quad hash table has two kinds of entries used for TCP traffic.

- Quad-touple entries carrying and using as input to hash function a quad of source and destination IP Addresses and source and destination TCP ports.
- Du-touple entries carrying and using as an input to the hash functions a pair of destination IP Address and destination TCP port.

The du-entries are allocated to filter out connection establishment packets that are targeting accelerated connections. Those entries are used to filter out TCP packets with a *SYN* bit set and *ACK* bit clear.

The quad-touple entries are allocated to filter out packets targeting already established accelerated QPs.

The passive side connection establishment flow:

- Listen.
  - — Allocate du-touple entry with local IP Address and local port socket is bound to.
- SYN is received on QP associated with du-touple.
  - — Allocate quad-touple entry with local/remote IP Addresses and local/remote TCP ports.
  - — Send SYN-ACK.
- ACK is received on the QP associated with quad-touple.

The active side connection establishment flow:

- Connect.
  - — Allocate quad-touple entry with local/remote IP Addresses and local/remote TCP ports.

  - — Allocate quad-touple entry with local/remote IP Addresses and local/remote TCP ports.
  - — Send SYN.
- SYN-ACK is received on QP associated with quad-touple.
  - — Send ACK.

UDA TCP uses the quad hash table in the same way this table is used for iWARP traffic. A hit in the table resolves a number of destination QPs. The E810 uses information in QP context to identify iWARP and UDA TCP traffic. If a packet misses the quad hash table, it is forwarded for the further processing to the stateless filters.

TCP UDA software should program the quad hash table with du-touple entry to establish UDA accelerated TCP connection (such as listen or connect socket calls) and to add a quad-touple entry once connection has been established. To avoid race and loss of data, the quad-touple entry can be allocated before the connection establishment handshake completed.

The E810 does not terminate TCP connections for UDA traffic. It forwards identified UDA TCP segments to the corresponding RQ. Multiple TCP tuples can be configured to be delivered to the same RQ. Each WQE in UDA TCP RQ is consumed by single TCP segment.

In addition to forwarding UDA TCP segments, the E810 provides several hardware accelerations.

- **TCP checksum calculation and validation** — The E810 supports calculation and validation of TCP checksum for UDA TCP segments. TCP packets with invalid a checksum is dropped and not delivered to RQ.

11.8.3 UDA Programming Interface

The UDA programming interface is based on the verb semantics described in Section 11.4.1. It uses same basic constructs as RDMA: QP, CQ, CEQ, AEQ and memory regions. Those constructs are described in detail in Section 11.4.1. In addition to those constructs, UDA uses address handles described later in this section.

11.8.3.1 CEQ

CEQ construct enables hardware to provide software with asynchronous completion notifications. Application has a full control over requesting asynchronous completion event for the particular CQ. UDA uses CEQ construct defined for RDMA, described in Section 11.4.1.2.

CEQs are allocated per MSI-X vector per PCI function, and if UDA and RDMA applications are deployed by the same PCI functions or the same application uses both types of traffic, it shares the same CEQ. Each entry in CEQ indicates that a requested CE was received by specified CQ and the application should poll CQ and retrieve completion of transmit or receive operation.

11.8.3.2 AEQ

AEQ construct enables hardware to report AE and error notifications. UDA uses AEQ defined for RDMA, described in Section 11.4.1.1. AEQ is allocated per PCI function and shared by iWARP and UDA if application(s) using both types of traffic are deployed in the same PCI function.

11.8.3.3 CQ

CQ is a construct that enables an application to receive notification about completed transmit and receive operations directly in the application address space. UDA uses CQ construct defined for RDMA, described in Section 11.4.1.3. CQ is associated with RQ and SQ to report completion of transmit and receive operations. The same CQ can be configured to report completion of transmit and receive operations on the same and different QPs. Application is allowed to use same CQ to report completion of UDA and RDMA traffic.

CQ entry indicates what operation has been completed, and depending on the completed operation, it carries additional information. The format of a UDA CQ entry is described in Section 11.8.5. Among other fields, CQ entry carries a 64-bit pointer to the QP completion information. This field can be used to identify a QP that posted completed operation.

Completion of receive operation is always reported in associated CQ. Completion of transmit operation is not necessarily reported and the application can control whether it wants to be notified about a completed transmit operation. Application MUST request completion of transmit operation at least once per SQ size worth of transmit requests.

11.8.3.4 QP

QP is a construct that enables an application to post transmit and receive operations directly from the application address space. QP consists of the pair of queues: SQ and RQ. UDA uses QP construct defined for RDMA, described in Section 11.4.1.5.

Unlike iWARP QP, UDA QP does not have to be associated with any particular connection or connection oriented service and enables mixing of various ULPs such as UDP and TCP on the same QP.

Userspace QP is associated with a particular local IP Address. Non-privileged UDA consumers need to allocate multiple UDA QPs to send UDA packets using different local IP Addresses. Local IP Address in QP context is used for UDA IP header generation only. E810 filters can be configured to receive packets targeting different local IP Addresses on the same QP.

Software should use WQEs described in Section 11.8.6 to post new transmit and receive work to the UDA QP.

UDA QP is associated with a particular TC, and VLAN. If the application needs to use multiple TCs and VLANs, it must create multiple QPs and spread traffic, respectively.

All software errors such as invalid WQE format, invalid STag and memory region boundary violation are considered to be critical errors and result in a transition of QP to the error state and immediately suspend transmit and receive operations of that QP. Respective AE would be reported via AEQ associated with QP (see Section 11.4.7).

11.8.3.5 Send Operation

The process of posting WRs to UDA SQ and flow is similar to one described for RDMA SQs in Section 11.4.1.5.2.

UDA QP enables the application to transmit individual Ethernet frames per posted SQ WQE. Each transmit WQE must refer to a single Ethernet frame. Software is responsible to limit the frame size to the configured MSS. If frame size exceeds MSS, frame would be discarded and error reported in CQE.

Software is allowed to post Ethernet frames shorter than a minimum Ethernet frame length. Hardware pads Ethernet frames to the minimal Ethernet frame length.

Hardware is responsible for Ethernet FCS and IP checksum generation. Software might request hardware to generate TCP/UDP checksum for non-fragmented datagrams.

In default operation mode, Ethernet and IP headers are generated by hardware using information provided in the address handle. Software can provide additional ULP header (e.g. UDP) either as a part of the application buffer referred by SQ WQE fragments or using a private header ring buffer (if enabled for the QP). If the size of the extended header exceeds the size of the private header ring buffer entry, the packet is discarded and the error is reported in CQE.

Privileged consumers are allowed to post Ethernet frames with all headers. To enable privileged SQ operation mode, the *Privileged Header Generation* bit should be set at QP creation (see Section 11.8.4).

Software can selectively request completion of WQEs by setting the *Signaled* bit. Completions with errors are returned regardless of the Signaled bit setting. UDA WQE is completed as soon as hardware finished processing WQE and validated lengths and fragments. Completion of UDA SQ WQE does not indicate that respective Ethernet frames are transmitted or received by the destination.

11.8.3.6 Receive Operation

The structure of RQ is a process of posting WRs to UDA RQ and flow is similar to one described for RDMA RQs in Section 11.4.1.5.8.

UDA QP enables an application to receive an individual Ethernet frame in pre-posted RQ WQEs. Each inbound Ethernet frame consumes one RQ WQE. Software needs to make sure that buffers posted to RQ WQE are large enough. If the size of the Ethernet frame exceeds the size of RQ WQE, the frame is truncated and an error is reported in CQE.

Ethernet frames failing CRC or checksum checks (IP/UDP/TCP) are dropped and not delivered to UDA QPs.

If an Ethernet frame is received while UDA RQ is empty, such frame is dropped and the GLPES_PFIP4RXDISCARD statistics counter is incremented.

RQ completion is delivered with each completed RQ WQE. If a packet carries one or more VLAN tags, the most inner VLAN tag is reported by RQ CQE. Note that VLAN tag is not stripped from the packet header.

11.8.3.7 Memory Registration

Memory region is a construct that enables hardware to access data directly from the application buffers. To enable direct hardware access, an application must register application buffers using the register memory region operation. UDA uses memory regions defined for RDMA described in Section 11.4.1.4.

Memory region construct is not associated with any particular traffic type, and registered memory region can be accessed both by RDMA and UDA traffic. This enables an application to have the same application buffer be directly accessed by iWARP and UDA traffic. For example, UDP data received via one of UDA QPs can be then accessed via RDMA QP without requiring copy operation.

UDA application must register application buffers to be used to send and receive UDA packets.

UDA supports only basic memory registration capabilities and does not support fast memory registration, bind and invalidate operations.

11.8.3.8 Address Handle

Address Handle is a construct that used solely by UD and UDA traffic. Address Handle allows protected generation of Ethernet and IP Headers for the UDA traffic. Application has to allocate Address Handle using CQP command described in Section 11.5.3.13 prior to posting data to transmit. Content of Address Handle is validated by driver. Allocated Address Handle is referred in transmit descriptors, and used by hardware to generate Ethernet and IP Headers.

Address Handle can be allocated by VF or PF driver. Standard anti-spoofing capabilities of the E810 internal switch applies to UDA traffic as well.

11.8.3.9 Push Mode Support

Push mode enables software to reduce processing latency of the short messages by pushing corresponding WQE with inline data to the memory-mapped adapter address space. Pushed WQE with inline data can be immediately processed by the adapter and transmitted to eliminate the need to read WQE and data from the host memory. See Section 11.4.1.5.6 for a detailed description of push mode concept and constructs.

UDA QPs take advantage of push mode defined for RDMA. Software can use push mode WQEs, described in Section 11.4.1.5.6 to reduce processing latency of the short UDA messages. Similar to RDMA, the size of the UDA push mode message is limited to 224 bytes. The UDA push mode message cannot use private ULP header buffers and the entire message with ULP headers must fit the push mode message size restrictions and posted as inline data. The remainder of push mode message software processing flow is identical to one described for RDMA messages in Section 11.4.1.5.6.

11.8.4 UDA QP Context Format

UDA QP context is used by software to initialize or update the E810's QP context. Create and Modify QP CQP operations (see Section 11.5.3.2) reference this structure.

Userspace QP is associated with a particular local IP Address. Non-privileged UDA consumers need to allocate multiple UDA QPs to send UDA packets using different local IP Addresses. The local IP Address in QP context is used for UDA IP header generation only. The E810 filters can be configured to receive packets targeting different local IP Addresses on the same QP.

Table 11-97. UDA QP Context Structure Format

Byte Offset	[Bit Range] Field Name			
0	[63:48] RSVD [47] Push_Mode_Enable [46:42] RSVD [41:32] Push_Page_Index [31] SQ_TPH_en [30] RQ_TPH_en [29] XMIT_TPH_en [28] RCV_TPH_en [27:22] RSVD [21:20] pd_index_high [19] RSVD [18:16] RSVD (dupack_thresh)		[15] RSVD (drop_out_of_order_seg) [14] RSVD [13:12] RSVD (limit) [11:10] RSVD [9:8] RQ_WQE_Size [7] RSVD (timestamp) [6] RSVD [5] RSVD (Insert_VLAN_Tag) [4] RSVD (NoNagle[1]) [3:2] RSVD [1:0] RSVD (iwarp_ddp_ver[2])	
8	[63:0] SQ_Address			
16	[63:0] RQ_Address			

Table 11-97. UDA QP Context Structure Format [continued]

Byte Offset	[Bit Range]	Field Name	[Bit Range]	Field Name
24	[63:48] [47:32] [31:24] [23]	RSVD (Dest_Port_Number) RSVD (Source_Port_Number) RSVD (Traffic_Class_or_TOS[2]) RSVD (avoid_stretch_ack)	[22:16] [15:12] [11:8] [7:0]	RSVD SQ_Size RQ_Size RSVD (Hop_Limit_or_TTL[2])
32	[63:32]	RSVD (Dest_IP_Address_1)	[31:0]	RSVD (Dest_IP_Address_0)
40	[63:32]	RSVD (Dest_IP_Address_3)	[31:0]	RSVD (Dest_IP_Address_2)
48	[63:48] [47:32] [31:30]	RSVD (ARP_Index) RSVD (VLAN_Tag[2]) RSVD	[29:16] [15:14] [13:0]	snd_mss RSVD RSVD (rcv_mss)
56	[63:48] [47:44] [43:40] [39:36] [35:32] [31:28]	pd_index RSVD RSVD (Snd_wscale) RSVD RSVD (Rcv_wscale) RSVD (TCP_state)	[27:24] [23] [22] [21] [20] [19:0]	RSVD RSVD (ignore_tcp_uns_options) RSVD (ignore_tcp_options) RSVD RSVD (wscale) RSVD (Flow Label[2])
64-128	[63:0]	RSVD		
136	[63:51] [50:32]	RSVD RxCmpQueueNum	[31:19] [18:0]	RSVD TxCmpQueueNum
144	[63:8] [7]	RSVD (Q2_Address) RSVD	[6:0]	Statistics_Instance_Index
152	[63:8]	RSVD	[7:0]	last_byte_sent
160	[63:57] [56:48] [47:41] [40:32] [31] [30] [29] [28] [27] [26]	RSVD RSVD (snd_mrk_offset) RSVD RSVD (rcv_mrk_offset) RSVD (rcv_no_mpa_crc) RSVD (assume_aligned_headers) RSVD (Receive_Markers) RSVD (iWARP_Mode) RSVD Use_Statistics_Instance	[25] [24] [23] [22] [21] [20] [19] [18:16] [15:8] [7:0]	PrivilegedEnable RSVD (FastRegisterEnable) RSVD (BindEnable) RSVD (Send_Markers) RSVD (rdmard_ok) RSVD (rdmawr_rdresp_ok) RSVD RSVD (IRD_Size) RSVD RSVD (ORD_Size)
168	[63:0]	QP_Completion_Context		
176	[63:44] [43:32] [31:26]	RSVD RSVD (Exception_UDA_Queue) RSVD	[25:16] [15:8] [7:0]	QS_Handle RQ_TPH_value SQ_TPH_value
184-248	[63:0]	RSVD		

1. This variable is a cached variable. For more details on cached context variables, see *Cached_Variables_Valid* in Modify QP operations (Section 11.5.3.2).
2. Only Valid for iWARP QPs.

PrivilegedEnable (1 bit)

This bit is used to enable privilege mode on STags.

Value	Description
0b	STag 0 is NOT allowed for a local STag, and an AEQE is generated with a privilege error indicated.
1b	A local STag of zero (STag 0) is allowed on this connection, and the TO field should be treated as a physical address.

XMIT_TPH_en (1 bit)

Value	Description
0b	THP is not used for data reads associated with this QP.
1b	TPH is enabled for data reads associated with this QP.

SQ_TPH_en (1 bit)

Value	Description
0b	THP is not used for this resource.
1b	TPH is enabled for the SQ of this QP.

SQ_TPH_value (8 bits)

If *SQ_TPH_en* is set (1b), TPH STag associated with SQ operations is initialized with *SQ_TPH_value*. If *SQ_TPH_en* is clear (0b), this field is ignored.

SQ_Size (4 bits)

This field encodes the maximum size for the WQ. The encoding of the SQ sizes are $4*2^{SQ\_Size}$ in terms of 32-byte quanta of memory. The following value are allowed for *SQ_Size*:

Value	Description
0001b	256 bytes
0010b	512 bytes
0011b	1024 bytes
0100b	2048 bytes
0101b	4096 bytes
0110b	8192 bytes
0111b	16384 bytes
1000b	32768 bytes
1001b	65536 bytes
1010b	131072 bytes
1011b	262144 bytes
1100b	524288 bytes
1101b	1048576 bytes
Note: All other values are reserved.	

Software can only allocate N-1 WQEs on the SQ, where N is a SQ size previously defined. Each WQE is variable in size and can consume up to 256 bytes of memory. For more information see Section 11.4.1.5.2. The minimum size for an SQ is four WQEs of the maximum size that is used.

RCV_TPH_en (1 bit)

Value	Description
0b	THP is not used for data placement associated with this QP.
1b	TPH is enabled for data placement associated with this QP.

RQ_TPH_en (1 bit)

Value	Description
0b	THP is not used for this resource.
1b	TPH is enabled for the RQ of this QP.

RQ_TPH_value (8 bits)

If *RQ_TPH_en* is set (1b), TPH STag associated with RQ operations is initialized with *RQ_TPH_value*. If *RQ_TPH_en* is clear (0b), this field is ignored.

RQ_Size (4 bits)

This field encodes the maximum size for the WQ. The encoding of the RQ sizes are $4*2^{RQ\_Size}$ in terms of 32-byte quanta of memory. The following value are allowed for *RQ_Size*:

Value	Description
0001b	256 bytes
0010b	512 bytes
0011b	1024 bytes
0100b	2048 bytes
0101b	4096 bytes
0110b	8192 bytes
0111b	16384 bytes
1000b	32768 bytes
1001b	65536 bytes
1010b	131072 bytes
1011b	262144 bytes
1100b	524288 bytes
1101b	1048576 bytes
Note: All other values are reserved.	

The actual number or WQEs that can be posted to the RQ is the size of the WQ divided by the WQE size determined from *RQ_WQE_Size*. Software can only submit N-1 WQEs to a WQ without processing completions for the WQ without exposing the possibility of a WQ overflow, where N is an RQ size previously defined. WQ overflow results in indeterministic behavior for the affected WQ. The minimum size for an RQ is four WQEs.

RQ_WQE_Size (2 bits)

Specifies the number of 32-byte chunks of memory included with each RQ WQE. The maximum number of additional fragments allowed for an RQ WQE is 13 for a total of 14 fragments and a maximum WQE size of 256 bytes. Valid values are:

Value	Description
00b	32 bytes per WQE (no additional fragments)
01b	64 bytes per WQE (1 or 2 additional fragments)
10b	128 bytes per WQE (3 to 6 additional fragments)
11b	256 bytes per WQE (7 to 13 additional fragments)

RxCmpQueueNum (19 bits)

This field specifies which of the 512K CQs is used for receive completion notification. The Rx and Tx completions can be mapped to the same CQ or different queues.

TxCmpQueueNum (19 bits)

This field specifies which of the 512K CQs is used for transmit completion notification. The Rx and Tx completions can be mapped to the same CQ or different queues.

SQ_Address (64 bits)

If *Virtual_WQs* bit is clear (0b), this field holds SQ base physical address. It must be aligned to an address divisible by 256 bytes. If *Virtual_WQs* bit is set (1b), this field specifies the first HMC PBLE index of the 1-level page list for the SQ (first *first_pm_pbl_index*).

RQ_Address (64 bits)

If *Virtual_WQs* bit is clear (0b), this field holds RQ base physical address. It must be aligned to an address divisible by 256 bytes. This is the RQ base pointer when this is an RDMA accelerated connection. If *Virtual_WQs* bit is set (1b), this field specifies the first HMC PBLE index of the 1-level page list for the SQ (first *first_pm_pbl_index*).

QS_Handle (10 bits)

This field specifies Tx-Scheduler queue set handle associated with the TC for this QP. See Section 8.3.3.4 for more information on scheduler configuration. For VFs, the *QS_Handle* is checked to ensure that the VF issuing the CQP command is associated with the *QS_Handle*.

pd_index (16 bits) and pd_index_high (2 bits)

Protection domain for this context. This specifies which one of the 32 KB UDA protection domains this connection context belongs too. There are no reserved index values.

QP_Completion_Context (64 bits)

This field is reported in CQEs and also in AEQEs.

Push_Page_Index (10 bits)

This field identifies the push page associated with the QP. For VFs, the push page index is function relative.

Push_Mode_Enable (1 bit)

This field indicates if push mode is enabled for the QP.

snd_mss (14 bits)

This field specifies a maximum MSS size allowed on that QP. MSS includes all headers and payload of the Ethernet frame, both generated by hardware and provided by software. If software posts a SQ WQE that exceeds this value, the frame is dropped, AE_UDA_XMIT_DGRAM_TOO_LONG AE is reported, and QP transitions to the error state.

Use_Statistics_Instance (1 bit)

This field indicates if the default per Private Memory Function statistics are used, or if one of the additional RDMA statistics instances are used for this QP.

Value	Description
0b	The default statistics are used.
1b	The statistics instance indicated by the *Statistics_Instance_Index* field are used.

Statistics_Instance_Index (7 bits)

This field specifies which of the additional RDMA statistics indexes are used if Use_Statistics_Instance is set (1b) This field is ignored if *Use_Statistics_Instance* is cleared (0b).

11.8.5 UDA CQ Entry Formats

The UDA CQE is the same as UD. See Section 11.7.1.1.

The UDA CQE is DEPRECATED.

11.8.6 UDA QP Completion Error Codes

UDA QP completion errors are typically reported via AEs with the E810. After fielding the AEs, software might issue the Flush WQES CQP operation (see Section 11.5.3.17) to complete any pending operations. Table 11-98 lists the completion codes reported by the E810 in a CQ entry (Table 11-91). Received packets that fail checksum validation are dropped by the chip and not delivered to RQ.

Table 11-98. UDA QP Error Codes

Major Error Code	Minor Error Code	Completion Reason	Description
0x0001	0x0001	WQE Flushed	The WQE has been flushed due to a ModifyQP state transition.
0x0002	0x0001	RQ WQE Too Short	The RQ WQE length was shorter than a length of received Ethernet frame. The frame was truncated, and an error reported.
0x0002	0x0003	SQ WQE Too Long	The SQ WQE length exceeded an MSS configured for the QP. SQ WQE length should include the size of the payload, hardware generated Ethernet and IP headers, and extended header provided by software.
0x0002	0x0004	SQ WQE Hdr Too Long	The SQ WQE extended header size exceeded the size of the private header buffer entry.

11.8.7 UDA QP Asynchronous Error Codes

UDA uses asynchronous errors to report unrecoverable critical errors; those errors mostly used to identify invalid use of UDA host interface. UDA uses the asynchronous error notification mechanism described in section Section 11.8.3.2. UDA shares asynchronous error codes with RDMA (see Section 11.4.7.

As a rule, once an unrecoverable critical error is detected on QP. QP transitions to the error state, and suspends all transmit and receive operations. Any incoming packet targeting a QP in an error state is silently discarded. To release buffers posted to a SQ and a RQ, software must use the Flush WQE CQP operation described in Section 11.5.3.17.

11.8.8 UDA Descriptor Formats

11.8.8.1 UDA SQ WQE Format - Send

The UDA SQE is the same as UD. See Section 11.7.1.3.3.

The UDA SQ entry below is DEPRECATED.

11.8.8.2 UDA SQ WQE Format - Send with Inline Data

The UDA send with inline data is the same as UD. See Section 11.7.1.3.4.

The UDA SQ entry is DEPRECATED.

11.8.8.3 UDA SQ WQE Format - NOP

The UDA NOP is the same as UD. See Section 11.7.1.3.2.

The UDA NOP entry is DEPRECATED.

11.8.8.4 UDA RQ WQE Format

The UDA RQE is the same as UD. See Section 11.7.1.4.

The UDA RQ entry is DEPRECATED.

11.9 Protocol Engine Statistics

11.9.1 Summary

Table 11-99 lists the RFCs relevant to the definition of PE statistics.

Table 11-99. Summary of MIBs Supported with E810 Hardware Statistics Counters

RFC	Description
1213	**Title:** Management Information Base for Network Management of TCP/IP-based Internets: MIB-II. **Status:** RFC1213 obsoletes RFC1158. RFC1213 is updated by RFC 2011(IP), 2012(TCP), 2013(UDP), 2863(Interfaces). There are MIB groups in RFC1213 that will be implemented by the E810 SNMP agent (such as the System group), but these do not require E810 hardware accelerations.
2011	**Title:** SNMPv2 MIB for the Internet Protocol using SMIv2. **Description:** Defines objects for managing implementations of IPv4 and ICMP. **Status:** Updates RFC1213, Obsoleted by RFC4293.
2012	**Title:** SNMPv2 MIB for the Transmission Control Protocol using SMIv2. **Description:** Defines objects for managing implementations of TCP. **Status:** Updates RFC1213, Obsoleted by RFC4022.
2013	**Title:** SNMPv2 MIB for the User Datagram Protocol using SMIv2. **Description:** Defines objects for managing implementations of UDP. **Status:** Updates RFC1213, Obsoleted by RFC4113.
4293	**Title:** MIB for the Internet Protocol (IP). **Description:** Defines objects for managing implementations of IPv4 and ICMP. **Status:** Obsoletes RFC2011, PROPOSED STANDARD.
4022	**Title:** MIB for the Transmission Control Protocol (TCP). **Description:** Defines objects for managing implementations of TCP. **Status:** Obsoletes RFC2012, PROPOSED STANDARD.
4113	**Title:** MIB for the User Datagram Protocol (UDP). **Description:** Defines objects for managing implementations of UDP. **Status:** Obsoletes RFC2013, PROPOSED STANDARD.

There are 128 sets of RDMA statistics. Each set can be assigned to an arbitrary/programmable group of one or more Queue Pairs. Any Queue Pair can be assigned to only one stat set. A typical configuration assigns one set to each active PCIe PF and VF, with the extra sets available for assignment as requested by the OS. For example, if 8 PCIe PFs and 32 PCIe VFs are active, then 40 RDMA stat sets are assigned to these functions, with the rest of the sets remaining for allocation as requested by the OS.

Stats colored Magenta are new for the E810.

The naming convention for the PE Statistics register is as follows:

- Physical function instance starts with GLPES_PF prefix following by the register name.

Table 11-100 lists PE registers by their physical function names. Virtual function register name can be recovered by substituting prefix.

Table 11-100. PE Statistics

Name	Size	MIB	Description
GLPES_PFRXVLANERR	24b	private	Ethernet received packets with incorrect VLAN_ID.
GLPES_PFIP4RXOCTS	48b	IP	IPv4 octets received.
GLPES_PFIP4RXPKTS	48b	IP	IPv4 packets received.
GLPES_PFIP4RXDISCARD	32b	IP	IPv4 packets received and discarded.
GLPES_PFIP4RXTRUNC	32b	IP	IPv4 packets received and truncated due to insufficient buffering space in UDA RQ.
GLPES_PFIP4RXMCPKTS	48b	IP	IPv4 multicast packets received.
GLPES_PFIP4RXMCOCTS	48b	IP	IPv4 multicast octets received.
GLPES_PFIP6RXOCTS	48b	IP	IPv6 octets received.
GLPES_PFIP6RXPKTS	48b	IP	IPv6 packets received.
GLPES_PFIP6RXDISCARD	32b	IP	IPv6 packets received and discarded.
GLPES_PFIP6RXTRUNC	32b	IP	IPv6 packets received and truncated due to insufficient buffering space in UDA RQ.
GLPES_PFIP6RXMCPKTS	48b	IP	IPv6 multicast packets received.
GLPES_PFIP6RXMCOCTS	48b	IP	IPv6 multicast octets received.
GLPES_PFIP4TXOCTS	48b	IP	IPv4 octets supplied by the PE to the lower layers for transmission.
GLPES_PFIP4TXPKTS	48b	IP	IPv4 packets supplied by the PE to the lower layers for transmission.
GLPES_PFIP4TXMCPKTS	48b	IP	IPv4 multicast packets transmitted.
GLPES_PFIP4TXMCOCTS	48b	IP	IPv4 multicast octets transmitted.
GLPES_PFIP6TXOCTS	48b	IP	IPv6 octets supplied by the PE to the lower layers for transmission.
GLPES_PFIP6TXPKTS	48b	IP	IPv6 packets supplied by the PE to the lower layers for transmission.
GLPES_PFIP6TXMCPKTS	48b	IP	IPv6 multicast packets transmitted.
GLPES_PFIP6TXMCOCTS	48b	IP	IPv6 multicast octets transmitted.
GLPES_PFIP4TXNOROUTE	24b	IP	IPv4 datagrams discarded due to routing problem (no hit in ARP table).
GLPES_PFIP6TXNOROUTE	24b	IP	IPv6 datagrams discarded due to routing problem (no hit in ARP table).
GLPES_PFTCPRXSEGS	48b	TCP	TCP segments received.
GLPES_PFTCPRXOPTERR	24b	Intel	TCP segments received with unsupported TCP options or TCP option length errors.
GLPES_PFTCPRXPROTOERR	24b	Intel	TCP segments received that are dropped by TRX due to TCP protocol errors.
GLPES_PFTCPTXSEG	48b	TCP	TCP segments transmitted.
GLPES_PFTCPRTXSEG	32b	TCP	Total number of TCP segments retransmitted.
GLPES_PFUDPRXPKTS	48b	UDP	UDP segments received without errors.
GLPES_PFUDPTXPKTS	48b	UDP	UDP segments transmitted without errors.
GLPES_PFRDMARXWRS	48b	Microsoft	RDMAP total RDMA write messages received.
GLPES_PFRDMARXRDS	48b	Microsoft	RDMAP total RDMA read request messages received.
GLPES_PFRDMARXSNDS	48b	Microsoft	RDMAP total RDMA send-type messages received.
GLPES_PFRDMATXWRS	48b	Microsoft	RDMAP total RDMA write messages sent.
GLPES_PFRDMATXRDS	48b	Microsoft	RDMAP total RDMA read request messages sent.

Table 11-100. PE Statistics [continued]

Name	Size	MIB	Description
GLPES_PFRDMATXSNDS	48b	Microsoft	RDMAP total RDMA send-type messages sent.
GLPES_PFRDMAVBND	48b	Microsoft	RDMA verbs total bind operations carried out.
GLPES_PFRDMAVINV	48b	Microsoft	RDMA verbs total invalidate operations carried out.
GLPES_PFRXNPECNMARKEDPKTS	56b		Packets with ECN bits indicating congestion.
GLPES_PFRXRPCNPHANDLED	32b		Counts the number of Congestion Notification Packets that have been handled by the reaction point.
GLPES_PFRXRPCNPIGNORED	24b		Counts the number of Congestion Notification Packets that have been ignored by the reaction point.
GLPES_PFTXNPCNPSENT	24b		Counts the number of Congestion Notification Packets that have been sent by the reaction point.

Note: A simple, conservative estimate for octet counter wrap times at 40 Gb/s:

- 32-bit counter wraps in 0.859 s.
- 36-bit counter wraps in 13.744 s.
- 40-bit counter wraps in 3.665 m.
- 48-bit counter wraps in 15.64 h.
- 56-bit counter wraps in 166.8 d.

There is a set of PE statistics instantiated only once. Table 11-101 lists these statistics, which are all defined by Intel and not part of a MIB.

Table 11-101. PE Intel-Specific Statistics Instantiated Once

Name	Size	Description
GLPES_RDMARXUNALIGN	32b	RDMA stat: TCP segments that probably have unaligned FPDUs.
GLPES_RDMARXMULTFPDUS	56b	RDMA stat: TCP segments that probably have multiple FPDUs.
GLPES_RDMARXOOONOMARK	32b	RDMA stat: FPDUs received out-of-order, with no MPA marker.
GLPES_RDMARXOOODDPLO	56b	RDMA stat: Number of out-of-order placed DDP segments.
GLPES_TCPRXPUREACKS	56b	TCP stat: Number of pure ACKs received.
GLPES_TCPRXONEHOLE	56b	TCP stat: Increments when an accelerated connection opens a first TCP hole.
GLPES_TCPRXTWOHOLE	56b	TCP stat: Increments when an accelerated connection opens a second TCP hole.
GLPES_TCPRXTHREEHOLE	56b	TCP stat: Increments when an accelerated connection opens a third TCP hole.
GLPES_TCPRXFOURHOLE	56b	TCP stat: Increments when an accelerated connection opens a fourth TCP hole.
GLPES_TCPTXRETRANSFAST	56b	TCP stat: Number of TCP re-transmits.
GLPES_TCPTXTOUTSFAST	56b	TCP stat: Number of TCP retransmission timeouts on connections attempting fast re-transmit.
GLPES_TCPTXTOUTS	56b	TCP stat: Number of TCP retransmission timeouts on connections that are not currently attempting fast re-transmit.

11.10 SR-IOV Protocol Engine Functionality

The E810 PE features are supported in virtualized operating systems that support SR-IOV direct assignment and IOMMUs. The E810 also supports para-virtualized drivers in virtualized operating system environments. The goal for the E810 PE programming model for VFs is to preserve as much of the non-virtualized programming model as possible to maximize the existing software investment. HMC resource profiles provide resource distribution/partitioning required to support this independent programming model in the VF. Backing pages for VF HMC pages for most HMC objects and page descriptor pages are allocated from the PF driver to provide improved security and stability. Backing pages for VF HMC PBL objects are allocated in the guest operating systems address space, the doorbell page from the VF BAR is mapped to guest userspace addresses and CQP operations are issued directly from the guest. The differences are in the programming model for VFs and show up in two areas:

- HMC objects — A HMC PCI function needs to be allocated for the VF and a few HMC objects are owned by the PF instead of the VF (specifically multicast groups and PE quad hash objects). Additionally, most HMC objects are allocated and managed by the PF driver. The VF driver uses the VF-to-PF mailbox or operating system-specific back channel mechanism to coordinate this activity.
- Interaction with the LAN portion of the driver.

To enable a VF driver to take advantage of PE functionality, an HMC PCI function must be allocated. There are 32 HMC VF PCI functions available in the E810. Firmware allocates HMC PCI functions on an as-needed basis between the PFs on a given adapter. This allocation takes place when the VF is allocated by the PF driver to a given guest operating system. In the case of multicast group HMC objects, these objects are always owned by PFs and require the VF driver to coordinate access to them by using VF-to-PF communication mechanisms. The difference in interaction with the LAN portion of the driver is operating system specific due to the different approaches that are taken on each vendor.

A sample initialization flow is as follows on top of the usual LAN VF initialization of interrupts, and so on:

1. When the PF driver created the PF CQP instance, it selected one of the SR-IOV HMC resource profiles to enable RDMA for VFs.
2. The PF driver (typically in the hypervisor or privileged host VM) allocates an HMC function on behalf of the VF using the PF CQP instance. See Section 11.5.3.10 for the WQE format.
3. The VF driver in the guest creates a new CQP instance. See Section 11.5.2.1.
4. The VF driver (or optionally the PF driver on the VF drivers behalf) issues a query FPM values operation. See Section 11.5.3.15.
5. The VF driver (or optionally the PF driver on the VF drivers behalf) issues a commit FPM values operation. See Section 11.5.3.16.
6. The PF driver allocates backing pages and page descriptors for all static HMC resources required for the VF and programs the SDs for the HMC function assigned to the VF using Update PE SDs operations. See Section 11.5.3.14. Note that backing pages are not allocated for PBLE HMC objects in the model, but page descriptors are. Also note that direct backing pages are not supported for VF PBLE HMC objects.
7. The VF driver (or optionally the PF driver on the VF drivers behalf) issues a static HMC pages allocated operation. See Section 11.5.3.14.
8. The PF driver (via communication from the VF driver) allocates backing pages and page descriptors for the initial set of HMC resources required by the VF driver. This step involves issuing one or more Update PE SD operations.
9. The VF driver uses its CQP instance to create CQ0, the CEQ, and AEQ for the VF.

10. During runtime, the VF driver signals the PF driver to allocate and initialize more HMC backing pages and/or PDs on its behalf. When PBLEs are needed during memory registration in the VF, the VF driver issues manage VF PBLE backing pages operations. See Section 11.5.3.12.

11.11 NVM RDMA Register Initialization

Table 11-102 describes the RDMA registers that are initialized via NVM settings.

Table 11-102. NVM Initial Values for RDMA Registers

Register	Field	Initial Value
GLPE_PEPM_CTRL	PEPM_PUSH_MARGIN	42
	PEPM_HALT	0
	PEPM_ENABLE	1
GLPE_TSCD_PEPM	MDQ_CREDITS	42
GLPE_PUSH_PEPM	MDQ_CREDITS	42

Table 11-103. Tx-Pipe Monitors Configuration in All Ports Setups

		Control Register	Negotiated Speed (Gb/s)					
			1 to 5 Port Configuration				6, 7, or 8 Port Configuration[1]	
			100	50	25	10[2]	25	10[2]
UPPER PIPE	Per Port Header Buffer Upper Pipe Monitors[3]	PRTDCB_TCUPM_REG_PE_HB_DTHR[0..7].*PORTOFFTH_L*	150	150	150	113	145	75
		PRTDCB_TCUPM_WAIT_PFC_PE_HB_DTHR[0..7].*PORTOFFTH_H*	300	300	300	226	290	150
		PRTDCB_TCUPM_WAIT_PE_HB_DTHR[0..31].*PORTOFFTH*	150	150	150	113	145	75
	Protocol Engine Pipe Monitor[4]	GLPE_PEPM_THRESH_I[0..511].*PEPM_PSQ_THRESH*	22	9	8	4	5	4
		GLPE_PEPM_THRESH_I[0..511].*PEPM_MDQ_THRESH*	492	420	336	124	288	124

1. If there are 6, 7, or 8 ports, the settings for 10 Gb/s and 25 Gb/s are changed to the values in the last two columns.
2. The 10G pipe monitor setup values apply to link speed 5G, 2.5G, 1G and 100M as well.
3. Programmed by EMP.
4. Programmed by CQP in the Protocol Engine.

Chapter 12 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 E810 in each platform configuration.

12.1 Features

Table 12-1 lists the manageability features in the LAN controller.

Table 12-1. 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. • 4 IPv4 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 1.1.0 - NC-SI Including ability to internally switch packets for communication between an OS and BMC. • DSP0261 1.2.0 - NC-SI over MCTP • DSP0236 1.3.0, DSP0238 1.0.1, DSP0237 1.1.0, DSP0239 1.4.0- MCTP over PCIe VDM and over SMBus • DSP0240 1.0.0 - PLDM Base • DSP0248 1.1.0, DSP0249 1.0.0- PLDM Monitoring and Control • DSP0267 1.0.1 - PLDM Firmware Update

12.2 Pass-Through Functionality

Pass-Through (PT) is the term used when referring to the process of sending and receiving Ethernet traffic over the sideband interface. The E810 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 12-1.

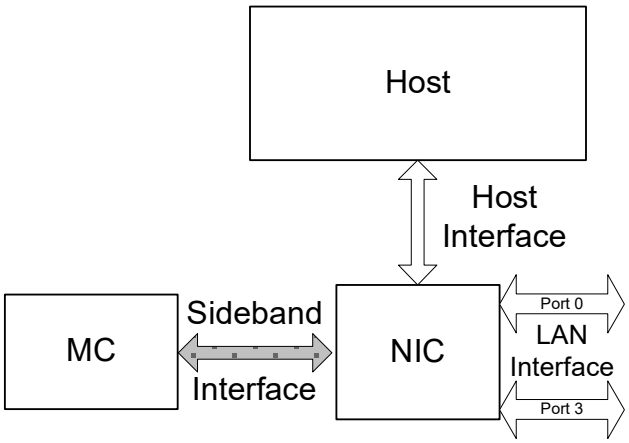


Figure 12-1. Sideband Interface

The sideband interface provides a mechanism by which the E810 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 E810 supports three sideband interfaces:

- SMBus (MCTP)
- NC-SI
- PCIe (together with MCTP) - when the system is up.

Notes:

- 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 E810 processing capabilities and can sustain any network bandwidth. The E810 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 STag, or VLAN tags. For traffic from the MC to the LAN, the maximum supported packet size is 1536 bytes including all tags.
- 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.

12.2.1 Supported Topologies

The E810 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 12.6).
- 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 12.7).
- 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 12.7).

The channel used is 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 6.3.57.2) and are common to all the ports in the device.

12.2.2 Pass-Through Packet Routing

When an Ethernet packet reaches the E810, it is examined and compared to a number of configurable filters. These filters are configurable by the MC and include, but are 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 12.4.

12.3 Components of the Sideband Interface

There are two components to a sideband interface:

- Physical layer
- Logical layer

12.3.1 Physical Layer

This is the electrical connection between the E810 and the MC.

12.3.1.1 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 (I^2C fast mode). The speed used while the E810 is the master of the bus is selected by the *SMBus Connection Speed* field in the SMBus Notification Timeout and Flags NVM word. When acting as a slave, the E810 can receive transaction with a clock running at up to 400 KHz.

12.3.1.1.1 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.

Table 12-2 lists the behavior of the E810 in each PEC configured mode for transactions directly handled by hardware after receiving packets with or without PEC.

Table 12-2. SMBus PEC Modes

SMBus transaction (relative to the E810)	E810 PEC Mode	Target PEC Mode[1]	
		PEC Enabled	PEC Disabled
Master 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.
Slave Read[3]	Enabled	(A) Target sends the PEC byte. PEC byte is ACKed by the slave.	(A) Target does not send PEC byte and generates stop afterwards.
	Disabled	(R) Target sends the PEC byte. PEC byte is NACKed by the slave	(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 transactions).
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 master, the transaction is still accepted and processed by the E810.

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 master SMBus write transactions. A PEC is added to slave read transactions and can be received in slave write transaction. If this bit is cleared, PEC is not added to master write or slave read transactions, a slave write transaction with PEC is dropped. This bit should always be set.

12.3.1.2 NC-SI over RBT

The E810 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.

12.3.1.2.1 Electrical Characteristics

The E810 complies with the electrical characteristics defined in the NC-SI 1.0.1 specification.

The E810's NC-SI behavior is configured at power-up in the following manner:

- The *Multi-Drop NC-SI* NVM bit (Section 6.3.57.3) defines the NC-SI topology (point-to-point or multi-drop; the default is point-to-point).

The E810 dynamically drives its NC-SI output signals (NC-SI_DV and NC-SI_RX) as required by the sideband protocol:

- At power-up, the E810 floats the NC-SI outputs.
- If the E810 operates in point-to-point mode, it starts driving the NC-SI outputs some time following power-up.
- If the E810 operates in a multi-drop mode, it drives the NC-SI outputs as configured by the BMC.

12.3.1.3 PCIe Vendor-Defined Messages (VDMs)

The E810 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.

12.3.2 Logical Layer

12.3.2.1 SMBus

12.3.2.1.1 SMBus Transactions

This section gives a brief overview of the SMBus protocol. Table 12-3 shows an example for a format of a typical SMBus transaction.

Table 12-3. Typical SMBus Transaction

1	7	1	1	8	1	8	1	1
S	**Slave 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 master for some transactions and a slave for others. The differences are identified in this document.

Shorthand field names are listed in Table 12-4 and are fully defined in the SMBus specification.

Table 12-4. 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)

12.3.2.1.2 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 E810's power up and resets.

12.3.2.1.3 SMBus ARP Functionality

The E810 supports the SMBus ARP protocol as defined in the SMBus 2.0 specification. The E810 is a persistent slave address device so its SMBus address is valid after power-up and loaded from the NVM. The E810 supports all SMBus ARP commands defined in the SMBus specification both general and directed.

SMBus ARP capability can be disabled through the NVM.

12.3.2.1.3.1 SMBus ARP Flow

SMBus ARP flow is based on the status of two flags:

- **Address Valid (AV)** — This flag is set when the E810 has a valid SMBus address.
- **Address Resolved (AR)** — This flag is set when the E810 SMBus address is resolved (SMBus address was assigned by the SMBus ARP process).

These flags are internal E810 flags and are not exposed to external SMBus devices.

Since the E810 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 E810 always has a valid SMBus address.

When the SMBus master 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 E810 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 E810 responds to SMBus ARP transactions that are issued by the master. The SMBus master issues a Get UDID command (general or directed) to identify the devices on the SMBus. The E810 always responds to the Directed command and to the General command only if its AR flag is not set.

After the Get UDID, the master assigns the E810 SMBus address by issuing an Assign Address command. The E810 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 E810 does not respond to the Get UDID General command. Note that all other commands are processed even if the AR flag is set. The E810 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 12-2 shows the E810 SMBus ARP flow.

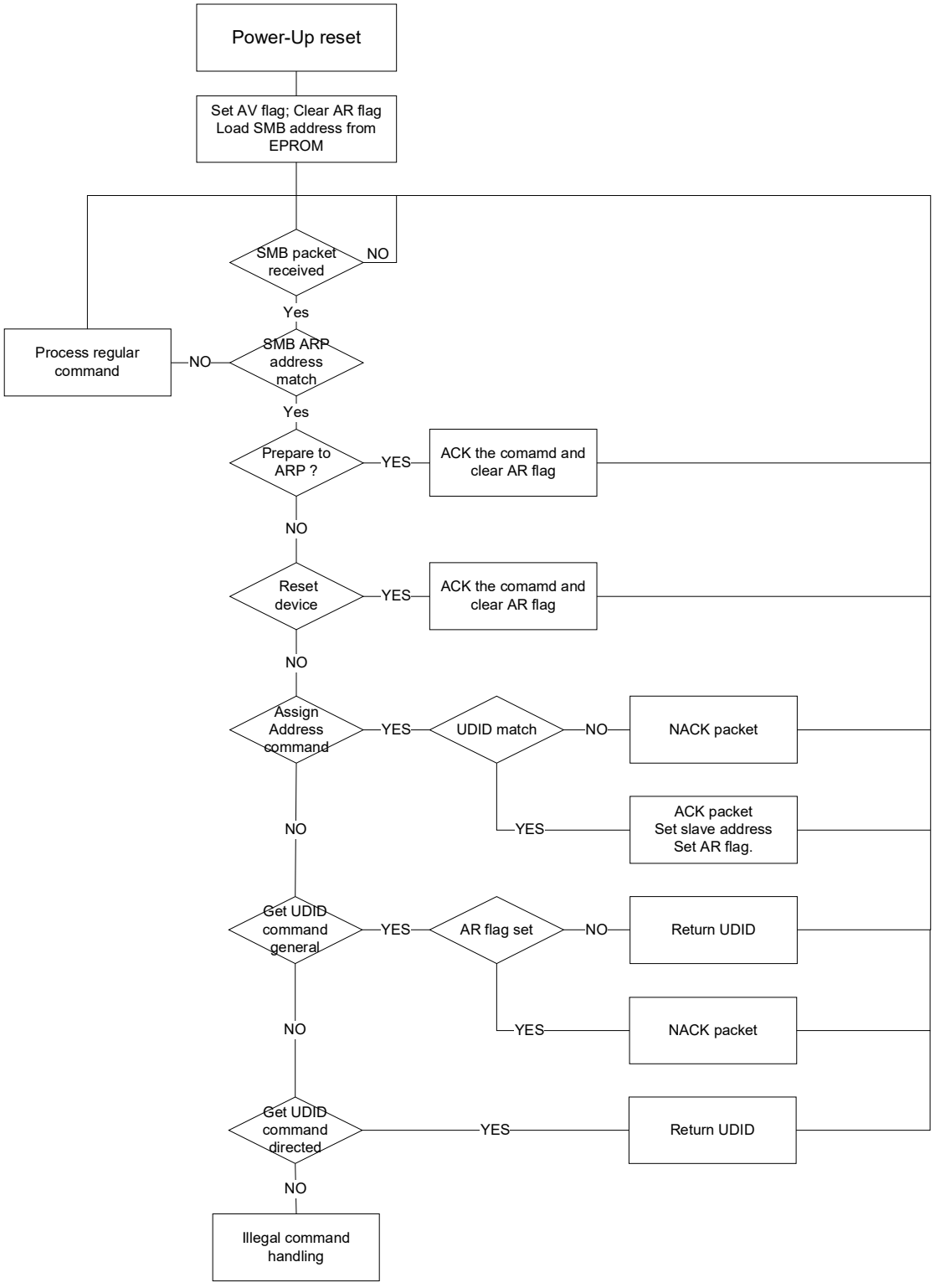


Figure 12-2. SMBus ARP Flow

12.3.2.1.3.2 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 12-5. 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	0x1590	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: 0x1590

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	0b/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/A1 in CAM1/CAM2 SKUs.	000b
B0 in CAM1/CAM2 SKUs. A0 in XXVAM2 SKU.	001b
C0	010b

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. Note that in the E810 there are up to eight MAC Addresses (one for each port).

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

12.3.2.2 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 E810 are discussed in Section 12.6.4. 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 12.3.1.2.
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 12.7.
3. MCTP over SMBus. As previously described, this layer is paired with the MCTP over PCIe to support Sx modes.

The E810 exposes one NC-SI package with eight channels, one per port. The E810 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.

12.3.2.2.1 Package ID Setting

The Package ID can be set either from the *Package ID* field in the NC-SI Configuration 1 NVM word (see Section 6.3.58.4) 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 6.3.58.5). The encoding is as follows:

00b = The Package ID is read from the *Package ID* field in the NC-SI Configuration 1 NVM word (Section 6.3.58.4) // Regular Card

01b = The the Package ID is (Package ID[2], SDP#1 value, SDP#0 value) // OCP, 2 SDPs.

10b = The package ID is (Package ID[2], SDP#1 value, Package ID[0]) // OCP single SDP

11b = Reserved.

Note: 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 are defined in the *PackageID SDP* field in the NC-SI Configuration 2 NVM word (Section 6.3.58.5).

12.3.2.2.2 Channel ID Mapping

The mapping of the channels to physical ports is according to the NC-SI Channel to Port Mapping NVM word (Section 6.3.58.15) if the *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 with channels yet.
For func = 0 to 7 { // 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 with 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.

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 then re-enumerates the package and discovers the new number of available channels.

12.4 Packet Filtering

Since both the host operating system and an MC use the E810 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 MC rather than the host operating system.

A high-level description of the host and MC filtering flow can be found in Section 7.8.2.

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 MC. This is the simplest filtering mechanism to use and it enables the MC to receive all types traffic (including, but not limited to, IPMI, NFS, HTTP, and so on).

The other type available uses a highly-configurable mechanism by which packets can be filtered using a wide range of parameters. Using this method, the MC can share a MAC Address (and IP Address, if desired) with the host operating system and receive only specific Ethernet traffic. This method is useful if the MC is only interested in specific traffic, such as IPMI packets.

12.4.1 Manageability Receive Filtering

This section describes the manageability receive packet filtering flow. Packet reception by the E810 can generate one of the following results:

- Discarded
- Sent to host memory
- Sent to the MC
- Sent to both the MC and host memory

The decisions regarding forwarding of packets to the host and to the MC are separate and are configured through two sets of registers. However, the MC might define some types of traffic as exclusive. This traffic is forwarded only to the MC, even if it passes the filtering process of the host. These types of traffic are defined using the PRT_MNG_MNGONLY register.

An example of packets that might be necessary to send exclusively to the MC might be specific TCP/UDP ports of a shared MAC Address or a MAC Address dedicated to the MC. If the MC configures the manageability filters to send these ports to the MC, 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 MC controls the types of packets that it receives by programming receive manageability filters. The following filters are accessible to the MC:

Table 12-6. Filters Accessible to MC

Filters	Functionality	When Reset
Filters Enable	General configuration of manageability filters.	LAN_PWR_GOOD
Manageability Only	Enables routing of packets exclusively to manageability.	LAN_PWR_GOOD
Manageability Decision Filters [7:0]	Configuration of manageability decision filters.	LAN_PWR_GOOD
MAC Address [3:0]	4 unicast MAC manageability addresses.	LAN_PWR_GOOD
VLAN Filters [7:0]	8 VLAN tag values.	LAN_PWR_GOOD
UDP/TCP Port Filters [15:0]	16 destination port values.	LAN_PWR_GOOD
Flexible 144 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
Special filters modifier	Used to define some special filtering options like 24-bit filtering of IPv6 Addresses and TCP/UDP selection of ASF ports.	LAN_PWR_GOOD

All filtering capabilities are available on the NC-SI interface. Part of the filters are programmed via standard NC-SI commands and part are programmed via the Intel OEM commands described in Section 12.6.4.

All filters are reset only on internal power-on-reset. Register filters that enable filters or functionality are also reset by firmware at internal firmware reset in NC-SI mode. These registers can be loaded from the NVM following a reset in SMBus mode. See Section 6.3.10 through Section 6.3.14 for descriptions of their location in the NVM map.

The high-level structure of manageability filtering is done using two or three steps.

1. The packet is routed by the switch as described in Section 7.8.7. If the switch determines the packet should be routed to the manageability VSIs, the next steps are taken.
2. The packet is parsed and fields in the header are compared to programmed filters.

3. A set of decision filters are applied to the result of the first step.

The following sections describe Step 2 and Step 3 previously listed.

Some general rules apply:

- Fragmented packets are passed to manageability but not parsed beyond the IP header.
- Packets with L2 errors (CRC, alignment, and so on) are not forwarded to the MC.
- Packets longer than 2 KB are filtered out.
- All the filters refers to the outer header of the packet and not to any tunneled header.

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 12.4.4.

12.4.2 L2 Filters

12.4.2.1 MAC and VLAN Filters

The manageability MAC filters allow a comparison of the destination MAC Address to one of four filters defined in the PRT_MNG_MMAH and PRT_MNG_MMAL registers.

The VLAN filters allow a comparison of the 12-bit VLAN tag to one of eight filters defined in the PRT_MNG_MAVTV registers. The VLAN tag compared is the innermost VLAN in the external L2 header.

12.4.2.2 EtherType Filters

Manageability L2 EtherType filters enable 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 (PRT_MNG_METF; up to four 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 example for using 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 E810 supports two 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 MC on the NC-SI link and the E810. Packets coming from the network are not expected to carry this EtherType, and such packets are blocked to prevent attacks on the MC. Flow control packets should be consumed by the MAC, and as such are not expected to be forwarded to the management interface.

Note: EtherType filters should not be configured with IPv4 or IPv6 EtherType values.

12.4.3 L3/L4 Filtering

The manageability filtering stage combines checks done at previous stages with additional L3/L4 checks to make a decision about whether to route a packet to the MC. The following sections describe the manageability filtering done at layers L3/L4 and final filtering rules.

12.4.3.1 ARP Filtering

The E810 supports filtering of ARP request packets (initiated externally) and ARP responses (to requests initiated by the MC).

To limit the reception of ARP packets to the ARP packets dedicated to this station (ARP target IP = MC IP), the ARP request/response filter can be bound to a specific IP Address by setting both the ARP Request/Response and the IP AND bits in an MDEF filter. Note that the IP bit is also set 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 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 value of 0x0800 (IPv4) should be added.

See Appendix A.5.3.8 for the format of ARP packets.

12.4.3.2 Neighbor Discovery Filtering and MLD

The E810 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 have 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.

See Appendix A.5.3.5 for the format of ICMPv6 packets.

12.4.3.3 RMCP Filtering

The E810 supports filtering by fixed destination port numbers: port 0x26F and port 0x298. These ports are IANA reserved for RMCP.

UDP or TCP protocols can be included in the comparison using the *PORT_26F_UDP*, *PORT_26F_TCP*, *PORT_298_UDP*, and *PORT_298_TCP* fields in the PRT_MNG_MSFM register.

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 RMCP traffic.

12.4.3.4 ICMP Filtering

The E810 supports filtering by ICMPv4. This filter matches if the IP *Protocol* field equals to 1b.

See Appendix A.5.3.4 for the format of ICMP packets.

12.4.3.5 Flexible Port Filtering

The E810 implements 16 flex source/destination port filters. The E810 directs packets whose L4 destination or source port matches to the MC. The MC must ensure that only valid entries are enabled in the decision filters.

For each flex port filter, filtering can be enabled for UDP, TCP or both. It can be enabled either on source or destination port.

12.4.3.6 IP Address Filtering

The E810 supports filtering by destination IP Address using IPv4 and IPv6 Address filters. These are dedicated to manageability. The E810 provides four IPv6 Address filters and four IPv4 Address filters.

For each IPv6 filter, the matching PRT_MNG_MSFM.*IPV6_n_MASK* bit defines if all the IP Address should be compared to the PRT_MNG_MIPAF6 register or only the 24 LS bits should be compared to the 24 LS bits of the PRT_MNG_MIPAF6 register.

The IPv4 match also rises for ARP packets for which the target IP matches the IP Address in the PRT_MNG_MIPAF4 register.

12.4.3.7 Checksum Filtering

The E810 might be instructed to direct packets to the MC 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 12.6.4.13.

Note: Checking the checksum of some complex IPv6 packets with routing extensions is not supported by the E810. Therefore, such packets are passed to the BMC even if their checksum is wrong and checksum offload is enabled. The BMC should check the checksum of such packets itself.

12.4.4 Flexible 144-Byte Filter

The E810 provides one flex TCO filter. This filter looks for a pattern match within the first 144 bytes of the packet.

The flex filter programming should ignore the presence of these fields.

If tags unknown to the programming instance (BMC) are expected in the packet (for example, if the device is removing STag before forwarding the packets to the BMC), the filter should be adapted accordingly by skipping/comparing the removed bytes in the filter.

Notes:

- The flex filter comparison should be disabled in the MDEF registers while the flex filter is being updated.
- The flexible filter is not applied to transmit packets and a transmit packet is considered as if it did not pass the filter.

The association of filters to ports is described in Table 12-7 according to the mode of operation, as defined by the GLGEN_MAC_LINK_TOPO register. For example, in a 4 ports mode, the filter to program for port 2 is PRT_MNG_FTFT_*_1_0.

Table 12-7. Number of Ports to Filter Mapping

Port Number/Mode	1/2 Ports Mode	4 Ports Mode	8 Ports Mode
0	PRT_MNG_FTFT_*_0_0 and PRT_MNG_FTFT_*_1_0[1]	PRT_MNG_FTFT_*_0_0	PRT_MNG_FTFT_*_0_0
1	PRT_MNG_FTFT_*_2_0 and PRT_MNG_FTFT_*_3_0[1]	PRT_MNG_FTFT_*_2_0	PRT_MNG_FTFT_*_2_0
2		PRT_MNG_FTFT_*_1_0	PRT_MNG_FTFT_*_1_0
3		PRT_MNG_FTFT_*_3_0	PRT_MNG_FTFT_*_3_0
4			PRT_MNG_FTFT_*_0_1
5			PRT_MNG_FTFT_*_2_1
6			PRT_MNG_FTFT_*_1_1
7			PRT_MNG_FTFT_*_3_1

1. The two registers should have the same value.

12.4.4.1 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. A packet below that length is not inspected. Valid values for this field are: 8*n, where n=1….
- **Data** — This is a set of up to 144 bytes comprised of values that header bytes of packets are tested against.
- **Mask** — This is a set of bit masks - one bit per byte of data in the packet that indicates if it is tested against its corresponding byte in the filter. Part of the bytes in the filter cannot be used for the filtering, so the mask is used to indicate which of the bytes are used for the filter.

12.4.4.2 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 follows:

Group	Test Bytes
0x0	0-29
0x1	30-59
0x2	60-89
0x3	90-119
0x4	120-127

   Each group of bytes must be configured using a separate command, where the group number is given as a parameter. The command has the following parameters:
   a. **Group number** — A 1-byte field indicating the current group addressed.
   b. **Data bytes** — Up to 30 bytes of test-bytes for the current group.

12.4.4.2.1 Flexible TCO Filter Configuration in NVM (Global MNG Offset 0x05)

This section describes the NVM module used to store the flex filter initial data in SMBus mode.

This module is a TLV in the PFA with a type of 7.

12.4.4.2.1.1 Section Header (Offset 0x00)

Bits	Name	Default	Description
15:0	Block Length	0xC	Section length in words (including CRC word and length word).

12.4.4.2.1.2 Flexible Filter Control (Offset 0x01)

Bits	Name	Default	Description
15	Last Filter		
14:8	Reserved		Reserved.
7	Apply Filter to LAN 7		
6	Apply Filter to LAN 6		
5	Apply Filter to LAN 5		
4	Apply Filter to LAN 4		
3	Apply Filter to LAN 3		

Bits	Name	Default	Description
2	Apply Filter to LAN 2		
1	Apply Filter to LAN 1		
0	Apply Filter to LAN 0		

12.4.4.2.1.3 Flexible Filter Length (Offset 0x02)

Bits	Name	Default	Description
15:7	Flexible Filter Length (bytes)		
6:0	Reserved		Reserved.

12.4.4.2.1.4 Flexible Filter Enable Mask (Offset 0x03-0x0A)

Bits	Name	Default	Description
15:0	Flexible Filter Enable Mask		

12.4.4.2.1.5 Flexible Filter Data - (Offset 0x0B-0x4A)

Bits	Name	Default	Description
15:0	Flexible Filter Data		

Note: This section loads all of the flexible filters. The control + mask + filter data are repeatable as the number of filters. Section length in offset 0 is for all filters.

12.4.4.2.1.6 Section Footer (Offset Block Length)

Bits	Name	Default	Description
15:8	CRC 8		CRC8 of the previous section.
7:0	Reserved		Reserved.

12.4.5 Configuring Manageability Filters

There are a number of pre-defined filters that are available for the MC to enable, such as ARPs and IPMI ports 0x298 and 0x26F. These are generally enabled by setting the appropriate bit within the PRT_MNG_MANC register using specific commands.

For more advanced filtering needs, the MC 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.

12.4.5.1 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 MC 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 is an ICMPv4 packet.
- 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 12-3. 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 one of the two registers. 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.

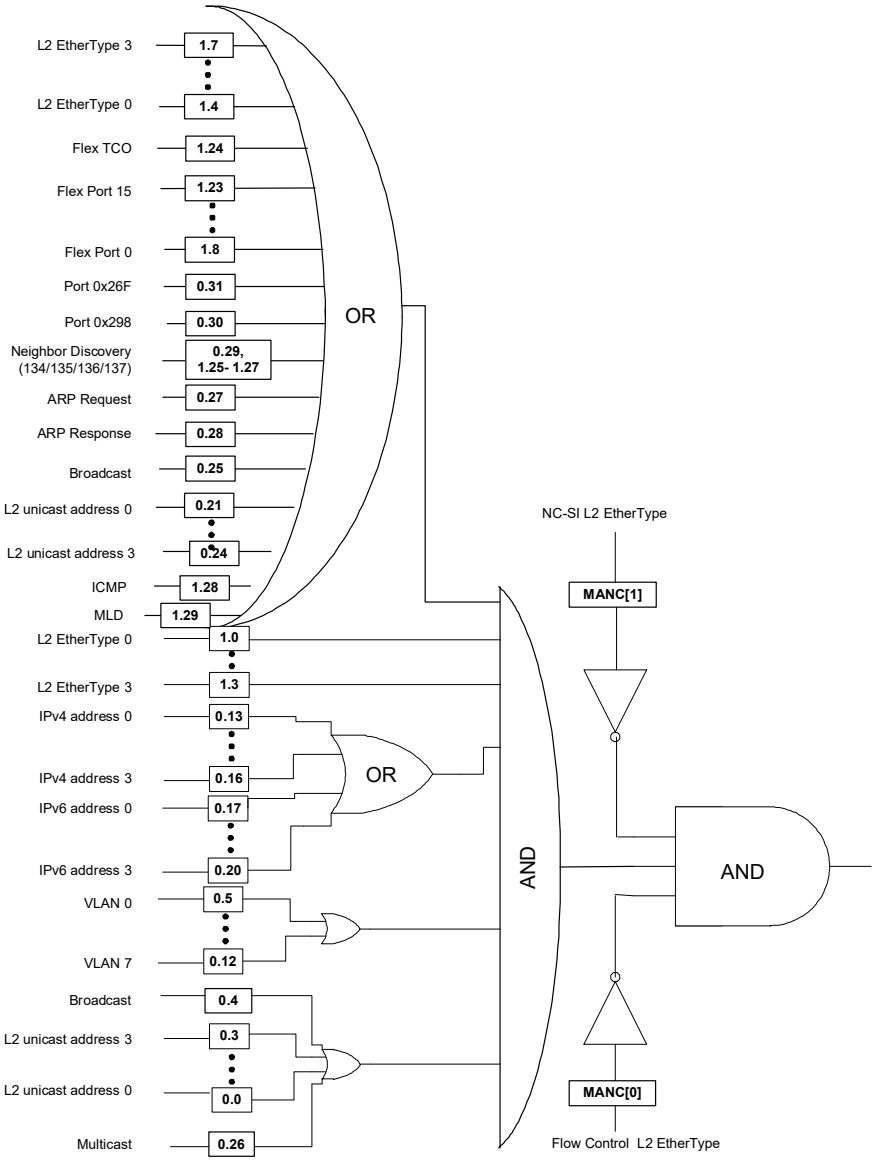


Figure 12-3. Manageability Decision Filters

A decision filter (for any of the eight filters) defines which of the previously described inputs are enabled as part of a filtering rule. The MC programs two 32-bit registers per rule (MDEF[7:0] and MDEF_EXT[7:0]) with the settings as described in Section 13.2.2.29.19 and Section 13.2.2.29.21. A set bit enables its corresponding filter to participate in the filtering decision.

In addition to the controls previously described, the PRT_MNG_MDEF_EXT.*APPLY_TO_HOST_TRAFFIC* and PRT_MNG_MDEF_EXT.*APPLY_TO_NETWORK_TRAFFIC* bits define which traffic is compared to this filter. At least one of these bits must be set for the filter to be valid.

If the PRT_MNG_MDEF_EXT.*APPLY_TO_HOST_TRAFFIC* bit is set, the traffic from the host is a candidate for this filter. If the PRT_MNG_MDEF_EXT.*apply_to_network_traffic* bit is set, the traffic from the network is a candidate for this filter. If both bits are set, this filter is applied to all traffic.

12.4.5.2 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 MC might 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 PRT_MNG_MNGONLY register (one bit per each of the eight decision rules) and the PRT_MNG_MDEF_EXT.*APPLY_TO_HOST_TRAFFIC* and PRT_MNG_MDEF_EXT.*APPLY_TO_NETWORK_TRAFFIC* bits.

Table 12-8 lists 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 12-8. Exclusive Traffic Behavior

Traffic Source	Filter Matches		Filter Does Not Match
	PRT_MNG_MNGONLY = 0	PRT_MNG_MNGONLY = 1	
From Network	Traffic is forwarded to 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 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 12.4.5.1) can be used as filters to pass traffic to the host.

Table 12-9. PRT_MNG_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.

PRT_MNG_MNGONLY is configurable when using NC-SI using the Set Intel Filters — Manageability Only command (see Section 12.6.4.5.2).

All manageability filters are controlled by the MC only and not by the LAN software device driver.

12.4.5.3 Global Controls

On top of the PRT_MNG_MDEF filters, the PRT_MNG_MANC registers contain some global controls applied to all the packets in order to be a 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 also set.
  - 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 untagged 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 relate to the inner VLAN.

Table 12-10 lists the relationship between the previously mentioned bits and the forwarding decisions:

Table 12-10. PRT_MNG_MANC Bits Impact

CASE\ PRT_MNG_MANC Bits	RCV_TCO_EN=0b	FIXED_NET_TYPE=1b and NET_TYPE!= What's in the Packet	EN_BMC2OS=0b (Assume EN_BMC2NET=1b)	EN_BMC2NET=0b (Assume EN_BMC2HOST=1b)
Packet sent from host and hits MDEF filters. (host-to-MC traffic)	Packet is not sent to the MC.	Packet is not sent to the MC.	Packet is not sent to the MC.	Packet is sent to the MC.
Packet sent from host and matches one of the EMP VSI. (host-to-EMP traffic).	Packet is sent to the EMP.	Packet is sent to the EMP.	Packet is sent to the EMP.	Packet is sent to the EMP.
Packet received from LAN and hits MDEF filters. (LAN-to-MC traffic)	Packet is not sent to the MC.	Packet is not sent to the MC.	Packet is sent to the MC.	Packet is not sent to the MC.
Packet received from LAN and matches on of the EMP VSI. (LAN-to-EMP traffic)	Packet is sent to the EMP.	Packet is sent to the EMP.	Packet is sent to the EMP.	Packet is sent to the EMP.
Packet sent from EMP and matches one of the host VSIs. (EMP-to-host traffic)	Packet is sent to the host (and optionally to the LAN).	Packet is sent to the host (and optionally to the LAN).	Packet is sent to the host (and optionally to the LAN).	Packet is sent to the host (and optionally to the LAN).
Packet sent from the EMP and does not match one of the host VSIs. (EMP-to-LAN traffic)	Packet is sent to the LAN.	Packet is sent to the LAN.	Packet is sent to the LAN.	Packet is sent to the LAN.
Packet sent from the MC and matches one of the host VSIs. (MC-to-host traffic)	Packet is sent to the host (and optionally to the LAN).	Packet is sent to the host (and optionally to the LAN).	Packet is sent to the LAN.	Packet is sent to the host (and optionally to the LAN).
Packet sent from the MC and does not match one of the host VSIs. (MC-to-LAN traffic)	Packet is sent to the LAN.	Packet is sent to the LAN.	Packet is sent to the LAN.	Packet is not sent to the LAN (optionally sent to host).

12.4.6 Filtering Programming Interfaces

The E810 provides multiple options to program the forwarding filters, depending on the interface used and the level of flexibility needed. The Table 12-11 lists the different options and points to the description of the relevant commands.

Table 12-11. 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 enable forwarding based on a dedicated MAC Address. The list of supported commands can be found in Section 12.6.2.1. When using these commands, one of the two other modes can be used to add finer grain filtering.
	Abstract (Shared MAC and IP)	The Intel OEM commands described in Section 12.4.6.1 and in Section 12.6.4.14 can be used to define which part of the shared MAC or shared IP traffic should be forwarded. When using these commands, the flexible filtering interface should not be used. This mode is activated using the *Set Shared Mode* command (Section 12.6.4.14.12.1)
	Flexible	This interface described in most of the sub-sections of Section 12.6.4. It 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.

12.4.6.1 Shared MAC and Shared IP Support

The E810 operates in systems where the same MAC and IP are shared between a platform's host operating system and its MC. To support such systems, the E810 supports additional shared MAC filtering options on top of what was supported in previous products. This section describes these options and the NC-SI commands used to program them.

Note: All filtering capabilities are exposed via the regular NC-SI packet reduction and packet addition commands and via the SMBus Set Filtering command. The interface described in this section is a more abstract NC-SI interface.

12.4.6.1.1 Sharing an IP and MAC Address

NC-SI over MCTP is used in desktop and mobile platforms. These platforms are typically used in enterprise environments outside of a data center. IP subnets in these environments are commonly designed such that more than 50% of their available addresses are assigned.

Hence, assigning a second IP Address to an MC would generally necessitate a subnet redesign. Instead, a single IP Address is typically shared between the host operating system and an MC in these platforms.

Because it is possible to bind multiple IP Addresses to a single MAC Address, the E810 needs to know the IP Address shared by an MC to deliver packets to it. An MC uses the Set IP Address command to communicate its IP Address to the E810. The Set IP Address command is defined in Section 12.6.4.14.1.

To notify the E810 that the MC intends to use a shared MAC, the Set Shared Mode command (Section 12.6.4.14.12.1) should be given before programming any filter using the regular NC-SI commands (Set MAC Address or Set VLAN) or the Intel OEM commands (Section 12.6.4.14).

12.4.6.1.1.1 TCP/UDP Ports Owned by an MC

A small subset of the TCP and UDP ports might be dedicated to an MC. The remaining ports are assigned to the host operating system. Hence, port-based filtering and the commands to configure it are required. For example, port-based filtering would be used to route WS-management packets to an MC.

The E810 needs to know the ports owned by an MC to deliver packets to it. An MC uses the Set Port command to communicate its ports to the E810. The Set Port command is defined in Section 12.6.4.14.3. The E810 supports 10 port filters.

The Set Binding command is used to define the combination of MAC, VLAN, IP and ports that should be met to forward packets to the MC (see Section 12.6.4.14.10 for more details).

12.4.6.1.1.2 Sharing Network Infrastructure Packets

In addition to management traffic, an MC needs to monitor network infrastructure traffic along with the host. For each flow, it is possible to define if it should include host traffic only, both host and network, or only network.

12.4.6.1.1.3 ARP Filters Enhancement

ARP request message filtering is controlled by the Enable Broadcast Filter command. However, as currently defined, this command causes either all or no ARP requests to go to an MC. For MCTP over SMBus, attempting to forward all ARP requests within a subnet to an MC can easily overwhelm the available bandwidth. Therefore, an option to have the E810 forward only ARP requests that contain a Target IP Address value that matches the IP Address used by an MC. An amendment to the Enable Broadcast Filter command is defined in the sections that follow to address this requirement.

12.4.7 Possible Configurations

This section describes ways of using management filters. Actual usage might vary.

12.4.7.1 Dedicated MAC Packet Filtering

1. Select one of the eight rules for dedicated MAC filtering.
2. Load the host MAC Address to one of the management MAC Address filters and set the appropriate bit in field 3:0 of the MDEF register.
3. Set other bits to qualify which packets are allowed to pass through. For example:
   - Set Bit 5 in the MDEF register to qualify with the first manageability VLAN.
   - Set relevant Bits 13 to 20 in the MDEF register to qualify with a match to one of the IP Addresses.
   - Set any L3/L4 bits (Bits 27 to 31 in the MDEF register and Bits 16 to 23 in MDEF_EXT) to qualify with any of a set of L3/L4 filters.

12.4.7.2 Broadcast Packet Filtering

1. Select one of the eight rules for broadcast filtering.
2. Set Bit 25 in the MDEF register of the decision rule to enforce broadcast filtering.
3. Set other bits to qualify which broadcast packets are allowed to pass through. For example:
   - Set Bit 5 in the MDEF register to qualify with the first manageability VLAN.
   - Set relevant Bits 13 to 20 in the MDEF register to qualify with a match to one of the IP Addresses.
   - Set any L3/L4 bits (Bits 27 to 31 in the MDEF register and Bits 16 to 23 in MDEF_EXT) to qualify with any of a set of L3/L4 filters.

12.4.7.3 VLAN Packet Filtering

1. Select one of the eight rules for VLAN filtering.
2. Set Bits 5 to 12 in the MDEF register to qualify with the relevant manageability VLANs.
3. 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 the MDEF register and Bits 16 to 23 in MDEF_EXT) to qualify with any of a set of L3/L4 filters.

12.4.7.4 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 or not 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 E810 supports IPv6 neighbor discovery protocol. Since the protocol relies on multicast packets, the E810 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 the 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.

12.4.7.5 Receive Filtering with Shared IP

When using the legacy SMBus interface or the MCTP interface, it is possible to share the host MAC and IP Address with an MC. This functionality is also available when using base NC-SI using Intel OEM commands.

When an MC 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:

1. Select one of the eight rules.
2. Set a manageability dedicated MAC filter to the host MAC Address and set the matching bit (0-3) in the MDEF register.
3. 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 an MC is responsible for handling the ARP protocol. Set Bit 27 or Bit 28 in the MDEF register for this functionality.

In NC-SI over MCTP, dedicated commands are used to enable shared IP filtering.

12.4.8 Determining Manageability MAC Address

If an MC needs to use a dedicated MAC Address or configure the automatic ARP response mechanism (only available in SMBus mode), it might be beneficial for an MC 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, an MC can read the system MAC Address and add one to it and be guaranteed of a unique MAC Address.

Note: Determining the IP Address being used by the host is beyond the scope of this document.

12.5 OS-to-BMC Traffic

12.5.1 Overview

Traditionally, the communication between a host and a local MC is not handled through the network interface and requires a dedicated interface, such as an IPMI KCS interface. The E810 enables the host and the local MC communication via the regular pass-through interface, and thus enable management of a local console using the same interface used to manage any MC in the network.

When this flow is used, the host sends packets to an MC through the network interface. The E810 examines these packets and then decides if they should be forwarded to an MC. On the inverse path, when an MC sends a packet on the pass-through interface, the E810 checks if it should be forwarded to the network, the host, or both. Figure 12-4 describes the flow for OS-to-BMC traffic for the NC-SI over RBT case. OS2BMC is also available when operating over MCTP.

The OS-to-BMC flow can be enabled using the *OS2BMC Enable* field for the relevant port in the OS-to-BMC configuration structure of the NVM.

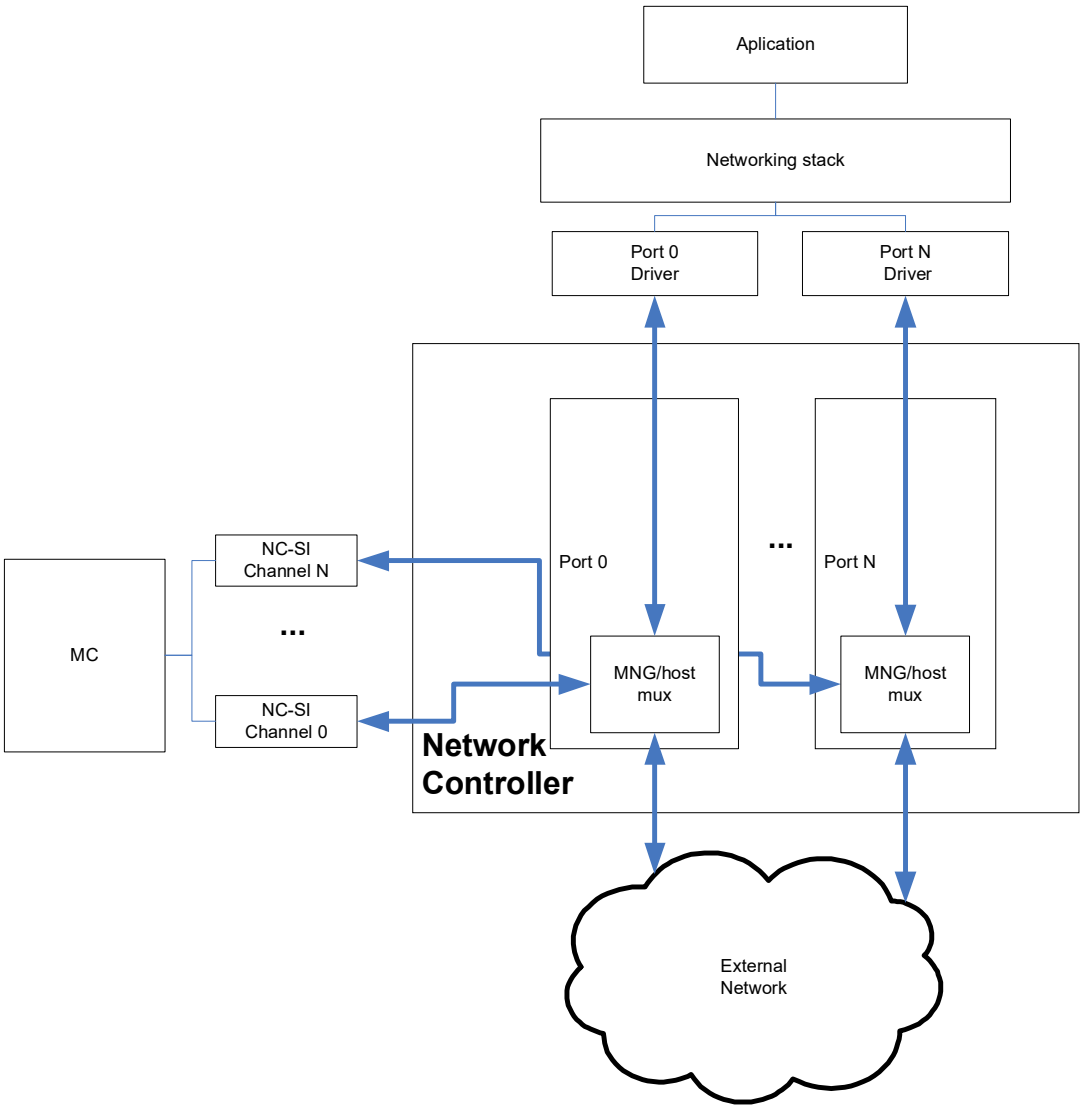


Figure 12-4. OS-to-BMC Flow

The OS-to-BMC flow is enabled only for ports enabled by the NC-SI Enable Channel command or via the *OS2BMC Enable* field for the relevant port in the OS-to-BMC configuration structure of the NVM.

OS-to-BMC traffic must comply with NC-SI specifications and is therefore limited to maximum sized frames of 1536 bytes (in both directions).

12.5.2 Filtering

12.5.2.1 OS-to-BMC Filtering

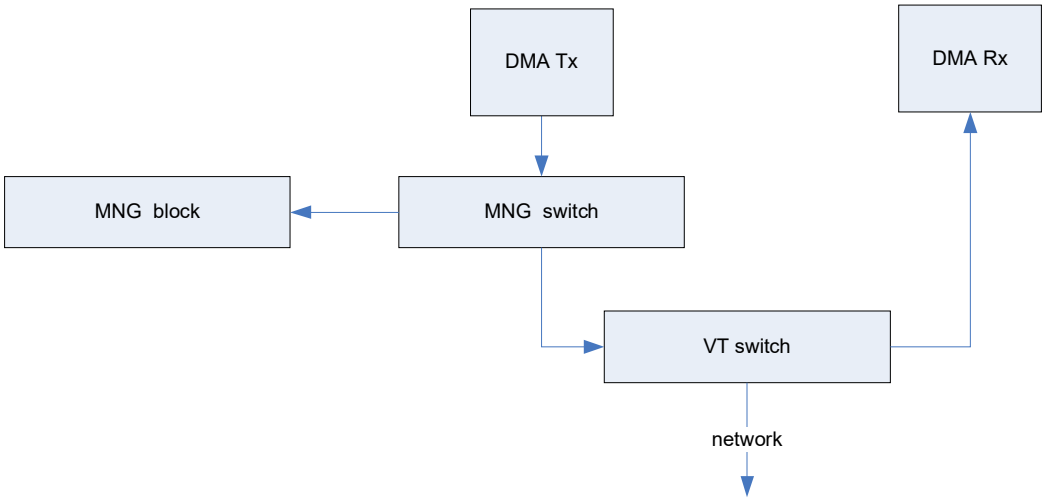


Figure 12-5. OS-to-BMC and VM-to-VM Filtering

12.5.2.2 BMC-to-OS Filtering

The flow used to filter packets from the BMC to the host is similar to the filtering of receive traffic.

Note: Traffic sent from the MC does not cause a PME event, even if it matches one of the wake-up filters set by the port.

12.5.3 Blocking Network-to-BMC Flow

In some systems the MC might have its own private connection to the network and might use a E810 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 PRT_MNG_MANC.*EN_BMC2NET* bit for the relevant port. The MC 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.

Note: The NC-SI channel should not be enabled for receive or transmit before at least one of the PRT_MNG_MANC.*EN_BMC2NET* or PRT_MNG_MANC.*EN_BMC2OS* fields is set, unless only used for AEN transmissions. In this case, the channel might be enabled for receive, but all receive filters should be cleared.

12.5.4 OS-to-BMC 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, then the host-to-MC 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 by manageability (either from the host or from the network) might be dropped if the manageability internal buffers are full.

12.5.5 Statistics

Packets sent from the operating system to the MC should be counted by all statistical counters as packets sent by the operating system. If they are sent to both the network and to the MC, they are counted once.

Packets sent from the MC 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.

See Section 9.6.5 for details of the statistics hierarchy.

12.5.6 OS-to-BMC Enablement

The E810 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 operating networking stack or the BMC stack, but 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, the operating system must avoid sending packets longer than 1.5 KB to the MC. Such packets are dropped.

12.5.7 SMBus Troubleshooting

This section outlines the most common issues found while working with PT using the SMBus sideband interface.

12.5.7.1 TCO Alert Line Stays Asserted After a Power Cycle

After the E810 resets, all its ports indicate a status change. If the MC only reads status from one port (slave address), the other one continues to assert the TCO alert line.

Ideally, the MC should use the ARA transaction to determine which slave asserted the TCO alert. Many customers only want to use one port for manageability, thus using ARA might not be optimal.

An alternative to using ARA is to configure part of the ports to not report status and to set its SMBus timeout period. In this case, the SMBus timeout period determines how long a port asserts the TCO alert line awaiting a status read from a MC; by default this value is zero (indicates an infinite timeout).

The SMBus configuration section of the NVM has a *SMBus Notification Timeout* (ms) field that can be set to a recommended value of 0xFF (for this issue). Note that this timeout value is for all slave addresses. Along with setting the *SMBus Notification Timeout* to 0xFF, it is recommended that the other ports be configured in the NVM to disable status alerting. This is accomplished by having the *Enable Status Reporting* bit set to 0b for the desired ports in the LAN configuration section of the NVM.

The third solution for this issue is to have the MC hard-code the slave addresses to always read from all ports. As with the previous solution, it is recommend that the other ports have status reporting disabled.

12.5.7.2 When SMBus Commands are Always NACK'd

There are several reasons why all commands sent to the E810 from a MC could be NACK'd. The following are most common:

- **Invalid NVM Image** — The image itself might be invalid or it could be a valid image and is not a PT image. As such, SMBus connectivity is disabled.
- **The MC is not using the correct SMBus address** — Many MC vendors hard-code the SMBus address(es) into their firmware. If the incorrect values are hard-coded, the E810 does not respond.
  - The SMBus address(es) can be dynamically set using the SMBus ARP mechanism.
- **The MC is using the incorrect SMBus interface** — The NVM might be configured to use one physical SMBus port. However, the MC is physically connected to a different one.
- **Bus Interference** — The bus connecting the MC and the E810 might be unstable.

12.5.7.3 SMBus Clock Speed is 16.6666 KHz

This can happen when the SMBus connecting the MC and the E810 is also tied into another device (such as an ICH) that has a maximum clock speed of 16.6666 KHz. The solution is to not connect the SMBus between the E810 and the MC to this device.

12.5.7.4 A Network-Based Host Application is Not Receiving Any Network Packets

Reports have been received about an application not receiving any network packets. The application in question was NFS under Linux. The problem was that the application was using the RMPC/RMCP+ IANA reserved port 0x26F (623) and the system was also configured for a shared MAC and IP Address with the operating system and the MC.

The management control to host configuration, in this situation, was setup not to send RMCP traffic to the operating system (this is typically the correct configuration). This means that no traffic send to port 623 was being routed.

The solution in this case is to configure the problematic application NOT to use the reserved port 0x26F.

12.5.7.5 Unable to Transmit Packets from the MC

If the MC has been transmitting and receiving data without issue for a period of time and then begins to receive NACKs from the E810 when it attempts to write a packet, the problem is most likely due to the fact that the buffers internal to the E810 are full of data that has been received from the network but has yet to be read by the MC.

Being an embedded device, the E810 has limited buffers that are shared for receiving and transmitting data. If a MC does not keep the incoming data read, the E810 can be filled up. This prevents the MC form transmitting more data, resulting in NACKs.

If this situation occurs, the recommended solution is to have the MC issue a Receive Enable command to disable more incoming data, read all the data from the E810, and then use the Receive Enable command to enable incoming data.

12.5.7.6 SMBus Fragment Size

The SMBus specification indicates a maximum SMBus transaction size of 32 bytes. Most of the data passed between the E810 and the MC over the SMBus is RMCP/RMCP+ traffic, which by its very nature (UDP traffic) is significantly larger than 32 bytes in length. Multiple SMBus transactions might therefore be required to move data from the E810 to the MC or to send a data from the MC to the E810.

Recognizing this bottleneck, the E810 handles up to 240 bytes of data in a single transaction. This is a configurable setting in the NVM. The default value in the NVM images is 32, per the SMBus specification. If performance is an issue, increase this size.

During initialization, firmware within the E810 allocates buffers based upon the SMBus fragment size setting within the NVM. E810 firmware has a finite amount of RAM for its use: the larger the SMBus fragment size, the fewer buffers it can allocate. Because this is true, MC implementations must take care to send data over the SMBus efficiently.

For example, E810 firmware has 3 KB of RAM it can use for buffering SMBus fragments. If the SMBus fragment size is 32 bytes, the firmware could allocate 96 buffers of size 32 bytes each. As a result, the MC could then send a large packet of data (such as KVM) that is 800 bytes in size in 25 fragments of size 32 bytes apiece. However, this might not be the most efficient way because the MC must break the 800 bytes of data into 25 fragments and send each one at a time.

If the SMBus fragment size is changed to 240 bytes, E810 firmware can create 12 buffers of 240 bytes each to receive SMBus fragments. The MC can now send that same 800 bytes of KVM data in only four fragments, which is much more efficient.

The problem of changing the SMBus fragment size in the NVM is if the MC does not also reflect this change. If a programmer changes the SMBus fragment size in the E810 to 240 bytes and then wants to send 800 bytes of KVM data, the MC can still only send the data in 32 byte fragments. As a result, firmware runs out of memory. This is because firmware created the 12 buffers of 240 bytes each for fragments; however, the MC is only sending fragments of size 32 bytes. This results in a memory waste of 208 bytes per fragment. Then when the MC attempts to send more than 12 fragments in a single transaction, the E810 NACKs the SMBus transaction due to not enough memory to store the KVM data.

In summary, if a programmer increases the size of the SMBus fragment size in the NVM (recommended for efficiency purposes) take care to ensure that the MC implementation reflects this change and uses that fragment size to its fullest when sending SMBus fragments.

12.5.7.7 Losing Link

Normal behavior for the Ethernet controller when the system powers down or performs a reset is for the link to temporarily go down and then back up again to re-negotiate the link speed. This behavior can have adverse affects on manageability.

For example, if there is an active FTP or Serial over LAN (SoL) session to the MC, this connection can be lost. To avoid this possible situation, the MC can use the Management Control command (detailed in Section 12.6.4.10) to ensure the link stays active at all times. This command is available when using the NC-SI sideband interface as well.

Care should be taken with this command, if the software device driver negotiates the maximum link speed, the link speed remains the same when the system powers down or resets. This can have undesirable power consumption consequences. Currently, when using NC-SI, the MC can re-negotiate the link speed. That functionality is not available when using the SMBus interface.

12.5.7.8 Enable Checksum Filtering

If checksum filtering is enabled, the MC does not need to perform the task of checking this checksum for incoming packets. Only packets that have a valid checksum is passed to the MC. All others are silently discarded.

This is a way to offload some work from the MC.

12.5.7.9 Still Having Problems?

If problems still exist, contact your field representative. Be prepared to provide the following:

- A SMBus trace if possible.
- A dump of the NVM image. This should be taken from the actual E810, rather than the NVM image provided by Intel. Parts of the NVM image are changed after writing (such as the physical NVM size).

12.6 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/

12.6.1 Overview

12.6.1.1 Terminology

The terminology in this section is taken from the NC-SI specification.

Table 12-12. NC-SI Terminology

Term	Definition
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.
Channel Arbitration	Refers 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 of hardware means.
Control Traffic/Messages/Packets	Command, response, and notification packets transmitted between BMC and the E810 for the purpose of managing NC-SI.
External Network Interface	The interface of the network controller that provides connectivity to the external network infrastructure (port).
Frame vs. Packet	Frame is used in reference to Ethernet, whereas packet is used everywhere else.
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.
Interface	This refers to the entire physical interface, such as both the transmit and receive interface between the management controller and the network controller.
Internal Host Interface	The interface of the network controller that provides connectivity to the host OS running on the platform.
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.
Management Controller (MC or 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, and so on).
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 master of the bus and communicates with multiple slave or target devices. In NC-SI, a management controller serves the role as the master, and the network controllers are the target devices.
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.

Table 12-12. NC-SI Terminology [continued]

Term	Definition
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.
Network Controller (NC)	The component within a system that is responsible for providing connectivity to the external Ethernet network world.
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.
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.
Pass-Through Traffic/Messages/Packets	Non-control packets passed between the external network and the BMC through the E810.
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 master/slave 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 master/slave relationship where the management controller is the master.
Remote Media	The capability to allow remote media devices to appear as if they were attached locally to the host.

12.6.1.2 System Topology

In NC-SI, each physical endpoint (NC package) can have several logical slaves (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 12-6 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.

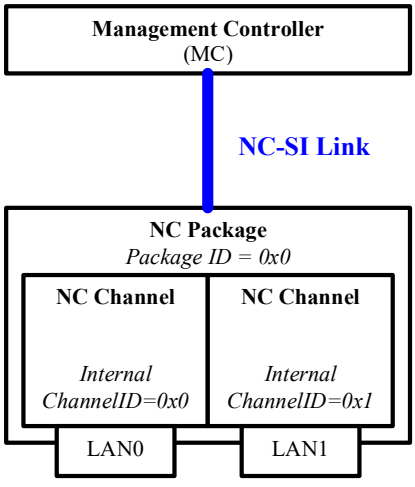


Figure 12-6. Single NC Package, Two NC Channels

Figure 12-7 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 12-7) mandate an arbitration mechanism.

The arbitration mechanism is described in Section 12.6.9.1.

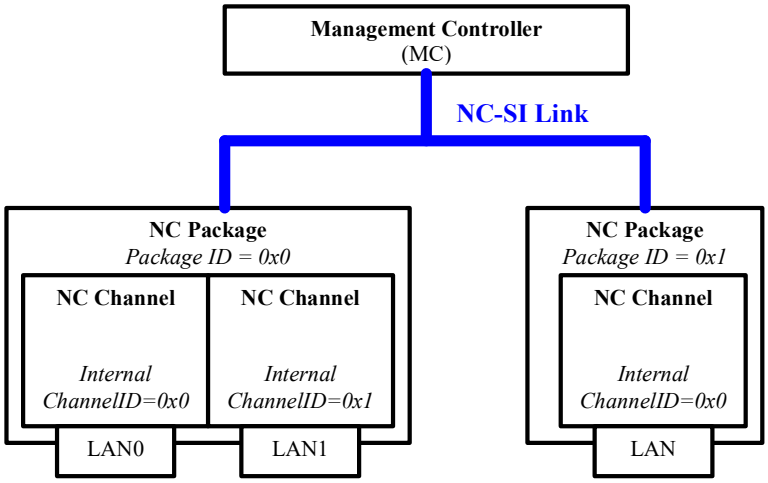


Figure 12-7. Two NC Packages (Left: with Two NC Channels, Right: with One NC Channel)

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 12.3.2.2.2 for details.

12.6.1.3 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:

- Control frames:
  - Configures and controls the interface.
  - Identified by a unique EtherType in their L2 header.
- Pass-through frames:
  - Actual LAN pass-through frames transferred from/to the MC.
  - Identified as not being a control frame.
  - Attributed to a specific NC channel by their source MAC Address (as configured in the NC by the MC).

12.6.1.3.1 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.

To be considered a valid command, a control frame must:

- Comply with the NC-SI header format.
- 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.
- Contain a correct payload checksum (if used).
- 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.

12.6.1.3.2 NC-SI Frames Receive Flow

Figure 12-8 shows the flow for frames received on the NC from the MC.

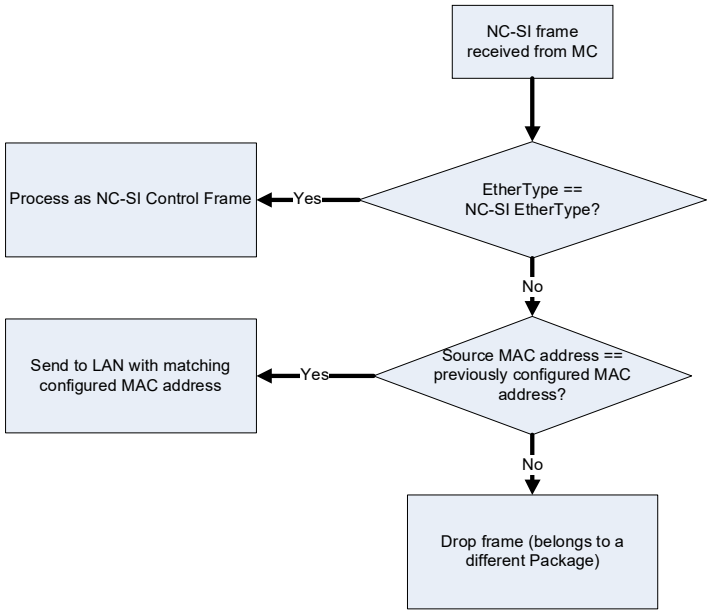


Figure 12-8. NC-SI Frames Receive Flow for the NC

12.6.2 NC-SI Standard Support

12.6.2.1 Supported Features

The E810 supports all the mandatory features of the NC-SI specification. Table 12-13 lists the support for standard NC-SI commands.

The E810 supports NC-SI 1.1.0.

Table 12-13. NC-SI Commands Support

Command	Support 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
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

Table 12-13. NC-SI Commands Support [continued]

Command	Support over RMII	Supported over MCTP with Pass-Through	Supported over MCTP without Pass-Through
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 E810 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 12.6.3.2 for details.
7. 7.This command is defined in DSP0222 NC-SI 1.2 draft.

Table 12-14 lists optional features supported and the level of support for partially supported commands.

Table 12-14. 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 one minute after actual driver change if the driver was taken down unexpectedly.
Get Controller Packet Statistics command	Yes (partially)	Supports the following counters[1]: 0-8, 11-16, 21-36[2]. The statistics 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 specification. 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	Supports all the counters[3]. ***Note:*** If a core reset occurs between reads of statistics, the reported statistics might be wrong.
Get NC-SI Pass-Through Statistics	Yes (partially)	Supports the following counters: 1, 6, and 7.
VLAN Modes	Yes (partially)	Supports only modes 1 and 3.
Buffering Capabilities	Yes	8 Kb
MAC Address Filters	Yes	Supports two MAC Addresses per port.
Channel Count	Yes	Supports eight channels.
VLAN Filters	Yes	Supports eight VLAN filters per port. Filtering ignores the *CFI* bit and the 802.1P priority bits.
Broadcast Filters	Yes	Support the following filters: • ARP • DHCP • Net BIOS

Table 12-14. Optional NC-SI Features Support [continued]

Feature	Implement	Details
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. 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.
2. As described in the Get Controller Packet Statistics Counter Numbers table in NC-SI specification.
3. The E810 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.

12.6.2.1.1 Error Conditions

If the device is not able to respond to a command within the 150 ms limit (three 50 ms retries), it responds with a response code of command unavailable (0x02) and a reason code of package not ready (0x04).

12.6.2.2 AEN Handling

Asynchronous events might occur when the device is not allowed to send them via the AEN Enable command. The following rules define the behavior of the E810 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 are 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), no AEN is sent.

12.6.2.2.1 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 a "Queue Shutdown" AQ command is received from the admit queue of the PF associated with this channel.

12.6.2.3 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 packet 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 12.7.7.1.3.
- 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.

12.6.3 External Link Control via NC-SI

12.6.3.1 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.

12.6.3.2 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 Specific Response (0x9) with a Set Link Host OS/Driver Conflict Reason (0x1).
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 support only AN, the AN is used to enforce a speed by negotiating only this speed. There are no modes supported by the E810 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 (0x9) with Link Command Failed-Hardware Access Error (0x6).

12.6.3.3 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 decide to use this feature, use it only if the link is down (trust the E810 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.

12.6.3.3.1 Set Link while LAN PCIe Functionality is Disabled

In cases where the E810 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 E810 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 renegotiated by the LAN device.

12.6.4 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 E810.

12.6.4.1 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 128 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 E810.
- Checksum offloading — Offloads IP/UDP/TCP checksum checking from the MC.
- OS2BMC control commands
- Firmware version commands
- Shared MAC and shared IP commands

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 (Section 12.6.4.1.1).

12.6.4.1.1 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			

12.6.4.1.2 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			

Notes:

- Responses have no command-specific reason code, unless otherwise specified within the command.
- The commands/responses described as follows include only the part up to the data. The padding and checksum are implied.

12.6.4.2 OEM Commands Summary

Table 12-15. 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.
		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 12-16 provides a summary of OEM commands.

Note: All the commands are supported over both RMII NC-SI and MCTP.

Table 12-16. OEM Commands Summary

Intel Command	Parameter	Command Name	Supported in MCTP without PT	Section Reference
0x00	0x00	Set IP Filters Control	No	12.6.4.3.1
0x01	0x00	Get IP Filters Control	No	12.6.4.4.1
0x02	0x0F	Set Manageability Only	No	12.6.4.5.2
	0x10	Set Flex Filter Mask and Length		12.6.4.5.3
	0x11	Set Flex Filter Data		12.6.4.5.4
	0x63	Set Flex TCP/UDP Port Filter		12.6.4.5.6
	0x64	Set Flex IPv4 Address Filter		12.6.4.5.7
	0x65	Set Flex IPv6 Address Filter		12.6.4.5.8
	0x67	Set EtherType Filter		12.6.4.5.9
	0x68	Set Packet Addition Extended Decision Filter		12.6.4.5.10
	0x69	Set Special Filter Modifiers		12.6.4.5.11

Table 12-16. OEM Commands Summary [continued]

Intel Command	Parameter	Command Name	Supported in MCTP without PT	Section Reference
0x03	0x0F	Get Manageability Only	No	12.6.4.6.2
	0x10	Get Flex Filter Mask and Length		12.6.4.6.3
	0x11	Get Flex Filter Data		12.6.4.6.4
	0x63	Get Flex TCP/UDP Port Filter		12.6.4.6.6
	0x64	Get Flex IPv4 Address Filter		12.6.4.6.7
	0x65	Get Flex IPv6 Address Filter		12.6.4.6.8
	0x67	Get EtherType Filter		12.6.4.6.9
	0x68	Get Packet Addition Extended Decision Filter		12.6.4.6.10
	0x69	Get Special Filter Modifiers		12.6.4.6.11
0x04	0x10	Set Extended Unicast Packet Reduction	No	12.6.4.7.2
	0x11	Set Extended Multicast Packet Reduction		12.6.4.7.3
	0x12	Set Extended Broadcast Packet Reduction		12.6.4.7.4
0x05	0x10	Get Extended Unicast Packet Reduction	No	12.6.4.8.1
	0x11	Get Extended Multicast Packet Reduction		12.6.4.8.2
	0x12	Get Extended Broadcast Packet Reduction		12.6.4.8.3
0x06	N/A	Get System MAC Address	Yes	12.6.4.9.1
0x20	N/A	Set Intel Management Control	No	12.6.4.10.1
0x21	N/A	Get Intel Management Control	No	12.6.4.11.1
0x22	N/A	Perform Intel TCO Reset	Yes	12.6.4.12.1
0x23	N/A	Enable IP/UDP/TCP Checksum Offloading	No	12.6.4.13.1
0x24	N/A	Disable IP/UDP/TCP Checksum Offloading	No	12.6.4.13.2
0x25	0x0	Set IP Address	No	12.6.4.14.1
	0x1	Get IP Address		12.6.4.14.2
	0x2	Set Port		12.6.4.14.3
	0x3	Get Port		12.6.4.14.4
	0x4	Enable Unicast Infrastructure Filter		12.6.4.14.5
	0x5	Get Shared IP Capabilities		12.6.4.14.6
	0x6	Shared IP Enable Broadcast Filtering		12.6.4.14.7
	0x7	Shared IP Enable Global Multicast Filtering		12.6.4.14.8
	0x8	Get Shared IP Parameters		12.6.4.14.9
	0x9	Set Binding		12.6.4.14.10
	0xA	Get Binding		12.6.4.14.11
	0xB	Set Shared Mode		12.6.4.14.12
0x40	0x01	Enable OS2BMC Flow	No	12.6.4.15.1
	0x02	Enable Network-to-BMC Flow		12.6.4.15.2
	0x03	Enable Both Network-to-BMC and Host-to-BMC Flows		12.6.4.15.3
	0x04	Set BMC IP Address		12.6.4.15.4

Table 12-16. OEM Commands Summary [continued]

Intel Command	Parameter	Command Name	Supported in MCTP without PT	Section Reference
0x41	N/A	Get OS2BMC Parameters	No	12.6.4.15.5
0x48	0x1	Get Controller Information	Yes	12.6.4.16.1
0x4B	N/A	Get ASIC Temperature	Yes	12.6.4.17.1

12.6.4.3 Set Intel Filters Control Commands

This command controls different aspects of the Intel filters.

12.6.4.3.1 Set IP Filters Control Command (Intel Command 0x00, Parameter 0x00)

	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	0x00	0x00	IP Filters Control (3-2)	
24...25	IP Filters Control (1-0)			

Where *IP Filters Control* has the following format.

Bit #	Name	Default	Description
0	IPv4/IPv6 Mode	1b	0b = IPv6 — There are zero IPv4 filters and four IPv6 filters. 1b = IPv4 — There are four IPv4 filters and four IPv6 filters.
1...31	Reserved		

Note: This command is kept for compatibility with other projects and has no effect in the E810.

12.6.4.3.1.1 Set 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		

12.6.4.4 Get Intel Filters Control Commands

12.6.4.4.1 Get IP Filters Control Command (Intel Command 0x01, Parameter 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		

12.6.4.4.1.1 Get 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...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 the E810.

12.6.4.5 Set Intel Filters Formats

12.6.4.5.1 Set Intel Filters Command (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	Manufacturer ID (Intel 0x157)			
20...21	0x02	Parameter Number	Filters Data (optional)	

12.6.4.5.1.1 Set Intel Filters 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	0x02	Filter Control Index	Return Data (optional)	

12.6.4.5.2 Set Manageability Only Command (Intel Command 0x02, Parameter 0x0F)

This command sets the PRT_MNG_MNGONLY register. The PRT_MNG_MNGONLY register controls whether pass-through packets destined to the MC are not forwarded to the Host operating system. The PRT_MNG_MNGONLY register is listed in Table 12-9.

	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	0x0F	Manageability Only (3-2)	
24...25	Manageability Only (1-0)			

12.6.4.5.2.1 Set Manageability Only 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	0x02	0x0F		

12.6.4.5.3 Set Flex Filter Mask and Length Command (Intel Command 0x02, 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	...	...	...	...
28...31	...	...	...	...
32...35	...	...	...	...
36...37	Mask Byte 15	Mask Byte 16	Reserved	
38	Length			

12.6.4.5.3.1 Set Flex Filter Mask and Length 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	0x02	0x10		

12.6.4.5.4 Set Flex Filter Data Command (Intel Command 0x02, 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	Manufacturer ID (Intel 0x157)			
20...	0x02	0x11	Filter Data Group	Filter Data 1
	...	Filter Data N		

The *Filter Data Group* parameter defines which bytes of the Flex filter are set by this command:

Table 12-17. 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

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.

12.6.4.5.4.1 Set Flex Filter Data 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	0x02	0x11		

12.6.4.5.5 Set Packet Addition Decision Filter Command (Intel Command 0x02, Parameter 0x61)

This command is no longer supported. Use the Set Packet Addition Extended Decision Filter Command (Intel Command 0x02, Parameter 0x68 instead (see Section 12.6.4.5.10).

12.6.4.5.6 Set Flex TCP/UDP Port Filter Command (Intel Command 0x02, 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	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x63	Port Filter Index	TCP/UDP Port MSB
24...25	TCP/UDP Port LSB	Port Flags		

Port Filter Index range: 0x0...0xA. When NC-SI 1.1 is supported, the range is 0x0...0x9.

Port Flags are as follows:

Bit 0: Match UDP ports
Bit 1: Match TCP ports
Bit 2: Match destination port (0) or source port (1)
Bit 7:3: Reserved

If flags are not present (payload length = 9), the match is done on TCP and UDP destination ports (legacy behavior).

If the *Port Filter Index* is larger than 10, a command failed Response Code is returned with Invalid Intel Parameter Number reason (0x5082).

12.6.4.5.6.1 Set Flex TCP/UDP Port Filter 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	0x02	0x63		

12.6.4.5.7 Set Flex IPv4 Address Filter Command (Intel Command 0x02, 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...23	0x02	0x64	IP Filter Index	IPv4 Address (3)
24...26	IPv4 Address (2-0)			

IP Filter Index range: 0x0...0x3.

12.6.4.5.7.1 Set Flex IPv4 Address Filter 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	0x02	0x64		

If the *IP Filter Index* is larger than 3, a command failed Response Code is returned with Invalid Intel Parameter Number reason (0x5082).

12.6.4.5.8 Set Flex IPv6 Address Filter Command (Intel Command 0x02, 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...23	0x02	0x65	IP Filter Index	IPv6 Address (MSB, byte 15)
24...27	...	...	...	...
28...31	...	...	...	...
32...35	...	...	...	...
36...38	...	...	IPv6 Address (LSB, byte 0)	

12.6.4.5.8.1 Set Flex IPv6 Address Filter 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	0x02	0x65		

If the *IP Filter Index* is larger the 3, a command failed Response Code is returned with Invalid Intel Parameter Number reason (0x5082).

12.6.4.5.9 Set EtherType Filter Command (Intel Command 0x02, 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 13.2.2.29.9.

12.6.4.5.9.1 Set EtherType Filter 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	0x02	0x67		

If the *EtherType Filter Index* is different than 2 or 3, a command failed Response Code is returned with Invalid Intel Parameter Number reason (0x5082).

12.6.4.5.10 Set Packet Addition Extended Decision Filter Command (Intel Command 0x02, Parameter 0x68)

See Figure 12-3 on page 1574 for a description of the decision filters structure.

The command must overwrite any previously-stored value. The value set is not checked.

	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	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 12-18.

Filter 1: See Table 12-19.

Table 12-18. Extended Decision Filter 0 Values

Bit #	Name	Description
3:0	Unicast (AND)	If set, packets must match unicast filter 0 to 3, respectively.
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.
24:21	Unicast (OR)	If set, packets can pass if match unicast filter 0 to 3, respectively or a different OR filter.
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 12-19. Extended Decision 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	ICMPv4 (OR)	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.

12.6.4.5.10.1 Set Packet Addition Extended Decision Filter 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	0x02	0x68		

If the *Extended Decision Filter Index* is larger than 5, a command failed Response Code is returned with Invalid Intel Parameter Number reason (0x5082).

12.6.4.5.11 Set Special Filter Modifiers Command (Intel Command 0x02, 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	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x69	Special Modifier Register MSB	
24...27	Special Modifier Register LSB		Padding	

Where the special modifier filter has the format as described in Section 13.2.2.29.17. The value set is not checked.

12.6.4.5.11.1 Set Special Filter Modifiers 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	0x02	0x69		

12.6.4.6 Get Intel Filters Formats

12.6.4.6.1 Get Intel Filters Command (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	Manufacturer ID (Intel 0x157)			
20...21	0x03	Parameter Number		

12.6.4.6.1.1 Get Intel Filters 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	0x03	Parameter Number	Return Data (optional)	

12.6.4.6.2 Get Manageability Only Command (Intel Command 0x03, Parameter 0x0F)

This command retrieves the PRT_MNG_MNGONLY register. The PRT_MNG_MNGONLY register controls whether pass-through packets destined to the MC are also be forwarded to the host operating system.

	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	0x0F		

12.6.4.6.2.1 Get Manageability Only Response

The PRT_MNG_MNGONLY register structure is listed in Table 12-9 on page 1575.

	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)			

12.6.4.6.3 Get Flex Filter Mask and Length Command (Intel Command 0x03, Parameter 0x10)

The following command retrieves the Intel flex filters mask and length. See Section 12.4.3.6 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		

12.6.4.6.3.1 Get Flex Filter Mask and Length 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	0x03	0x10	Mask Byte 1	Mask Byte 2
28...31	...	...	...	...
32...35	...	...	...	...
36...39	...	...	...	...
40...43	...	Mask Byte 16	Reserved	
44	Flexible Filter Length			

12.6.4.6.4 Get Flex Filter Data Command (Intel Command 0x03, Parameter 0x11)

The following command retrieves the Intel flex filters data.

	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	0x11	Filter Data Group 0...4	

The *Filter Data Group* parameter defines which bytes of the flex filter are returned by this command.

Table 12-20. 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

12.6.4.6.4.1 Get Flex Filter Data 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...	0x03	0x11	Filter Group Number	Filter Data 1
	...	Filter Data N		

12.6.4.6.5 Get Packet Addition Decision Filter Command (Intel Command 0x03, Parameter 0x61)

This command is no longer supported. Use the Get Packet Addition Extended Decision Filter Command (Intel Command 0x02, Parameter 0x68 instead (see Section 12.6.4.6.10).

12.6.4.6.6 Get Flex TCP/UDP Port Filter Command (Intel Command 0x03, 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	Manufacturer ID (Intel 0x157)			
20...22	0x03	0x63	TCP/UDP Filter Index	

TCP/UDP Filter Index range: 0x0...0x2.

12.6.4.6.6.1 Get Flex TCP/UDP Port Filter 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	0x03	0x63	TCP/UDP Filter Index	TCP/UDP Port (1)
28...29	TCP/UDP Port (0)	Port Flags		

TCP/UDP Filter Index range: 0x0...0x2.

12.6.4.6.7 Get IPv4 Address Filter Command (Intel Command 0x03, 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: *IPv4 Filter Index* range: 0x0...0x3.

IPv6 Mode: This command should not be used in IPv6 mode.

12.6.4.6.7.1 Get IPv4 Address Filter 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	0x03	0x64	IPv4 Filter Index	IPv4 Address (3)
28...30	IPv4 Address (2-0)			

12.6.4.6.8 Get IPv6 Address Filter Command (Intel Command 0x03, 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 filter 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.

12.6.4.6.8.1 Get IPv6 Address Filter 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	0x03	0x65	IPv6 Filter Index	IPv6 Address (MSB, Byte 16)
28...31	...	...	...	...
32...35	...	...	...	...
36...39	...	...	...	...
40...42	...	...	IPv6 Address (LSB, Byte 0)	

12.6.4.6.9 Get EtherType Filter Command (Intel Command 0x03, 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...22	0x03	0x67	EtherType Filter Index	

Valid indices: 0...3

12.6.4.6.9.1 Get EtherType Filter 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	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 with Invalid Intel Parameter Number reason (0x5082).

12.6.4.6.10 Get Packet Addition Extended Decision Filter Command (Intel Command 0x03, Parameter 0x68)

This command enables the MC to retrieve the extended decision filter.

	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	0x68	Extended Decision Filter Index	

12.6.4.6.10.1 Get Packet Addition Extended Decision Filter 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	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 with Invalid Intel Parameter Number reason (0x5082).

12.6.4.6.11 Get Special Filter Modifiers Command (Intel Command 0x03, 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	Manufacturer ID (Intel 0x157)			
20...23	0x03	0x69	Padding	

Where the special modifier filter has the format as described in Section 13.2.2.29.17.

12.6.4.6.11.1 Get Special Filter Modifiers 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	0x03	0x69	Special Modifier Register MSB	
28...31	Special Modifier Register LSB		Padding	

12.6.4.7 Set Intel Packet Reduction Filters Formats

The non-extended commands are obsolete. The extended commands (Section 12.6.4.7.2 to Section 12.6.4.7.4) should be used instead.

12.6.4.7.1 Set Intel Packet Reduction Filters Command (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	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 12-21. Packet Reduction Field Description

Bits	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.
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 12-22. Extended Packet Reduction Field Description

Bits	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.
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 12-21 and Bits 3:2 in Table 12-22 work in an AND manner; it must be true for a packet to pass (if was set).
- Bits 28 in Table 12-21 and Bits 24:10 in Table 12-22 work in an OR manner; at least one of them must be true for a packet to pass (if any were set).

12.6.4.7.1.1 Set Intel Packet Reduction Filters 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	0x04	Packet Reduction Index		

12.6.4.7.2 Set Extended Unicast Packet Reduction Command (Intel Command 0x04, 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	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 12-21 and Table 12-22 for description of the Unicast Extended Packet Reduction format.

12.6.4.7.2.1 Set Extended Unicast Packet Reduction 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	0x04	0x10		

12.6.4.7.3 Set Extended Multicast Packet Reduction Command (Intel Command 0x04, 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	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		

The command overwrites any previously-stored value.

Note: See Table 12-21 and Table 12-22 for description of the Multicast Extended Packet Reduction format.

12.6.4.7.3.1 Set Extended Multicast Packet Reduction 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	0x04	0x11		

12.6.4.7.4 Set Extended Broadcast Packet Reduction Command (Intel Command 0x04, Parameter 0x12)

	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	0x12	Extended Broadcast Reduction Filter MSB	...
24...27	...	Extended Broadcast Reduction Filter LSB	Broadcast Reduction Filter MSB	...
28...29	...	Broadcast Reduction Filter LSB		

The command overwrites any previously-stored value.

Note: See Table 12-21 and Table 12-22 for description of the Broadcast Extended Packet Reduction format.

12.6.4.7.4.1 Set Extended Broadcast Packet Reduction 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	0x04	0x12		

12.6.4.8 Get Intel Packet Reduction Filters Formats

The non-extended commands are obsolete. Use the extended commands (Section 12.6.4.8.1 to Section 12.6.4.8.3) instead.

12.6.4.8.1 Get Extended Unicast Packet Reduction Command (Intel Command 0x05, 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	Manufacturer ID (Intel 0x157)			
20...21	0x05	0x10		

12.6.4.8.1.1 Get Extended Unicast Packet Reduction 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	0x05	0x00	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)			

12.6.4.8.2 Get Extended Multicast Packet Reduction Command (Intel Command 0x05, 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	Manufacturer ID (Intel 0x157)			
20...21	0x05	0x11		

12.6.4.8.2.1 Get Extended Multicast Packet Reduction 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	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)			

12.6.4.8.3 Get Extended Broadcast Packet Reduction Command (Intel Command 0x05, Parameter 0x12)

	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	0x12		

12.6.4.8.3.1 Get Extended Broadcast Packet Reduction 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	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)			

12.6.4.9 System MAC Address

12.6.4.9.1 Get System MAC Address Command (Intel Command 0x06)

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 Address 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.

	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	0x06			

12.6.4.9.1.1 Get System MAC Address 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	0x06	MAC Address		
28...30	MAC Address			

12.6.4.10 Set Intel Management Control Formats

12.6.4.10.1 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:

Bits	Default	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.
7:1	0x0	Reserved.

Caution: The System Firmware Update Control option in the command should be supported only over MCTP over SMBus.

Notes:

- 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.
- If the parameter that the BMC sets is not supported by the device, the device returns Response Code 0x1 (Command Failed) with Reason Code 0x5082 (Invalid Intel command parameter number). The setting of the *System Firmware Update Control* field should be kept across EMP reset, so it should be kept in the Alternate RAM.
- The policy set by this command is applicable for any firmware update channel, either from host or via PLDM firmware update.

12.6.4.10.1.1 Set Intel Management 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	0x20	0x00		

12.6.4.11 Get Intel Management Control Formats

12.6.4.11.1 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		

12.6.4.11.1.1 Get Intel Management 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...26	0x21	0x00	Intel Management Control 1	

Where *Intel Management Control 1* is as described in Section 12.6.4.10.1.

12.6.4.12 TCO Reset

Depending on the bit set in the *TCO Mode* field, this command causes the E810 to perform either:

1. **TCO Reset**
   - If *Force TCO Reset* is enabled in the NVM (see Section 6.3.57.2), the TCO Reset command clears the data path (Rx/Tx) of the E810 to enable the MC to transmit/receive packets through the E810.
   - 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 internal driver-up indication. If TCO reset is disabled in the NVM, *Force TCO Reset* 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 6.3.57.3), the TCO Isolate command disables PCIe write operations to the LAN port.
   - If *TCO Isolate* is disabled in NVM, the E810 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.
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.

Notes: 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 a firmware reset, the MC needs to re-initialize all ports. A firmware reset causes a global reset of the entire device (GLOBR).

Only one of the fields should be set in a given command. Setting more than one field might yield unexpected results.

12.6.4.12.1 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	Bits	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** Reload 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 NVM is loaded. ***Note:*** A reset of the internal firmware causes a reset of the entire device.
Reserved	7:3	Reserved. Set to 0x00.

1. TCO isolate host write operation enabled in NVM.

Note: For compatibility, the TCO Reset command without the *TCO Mode* parameter is accepted (TCO reset is done).

12.6.4.12.1.1 Perform Intel TCO Reset 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	0x22			

12.6.4.13 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.

12.6.4.13.1 Enable IP/UDP/TCP 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			

12.6.4.13.1.1 Enable IP/UDP/TCP Checksum Offloading 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	0x23			

12.6.4.13.2 Disable IP/UDP/TCP Checksum Offloading Command (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	Manufacturer ID (Intel 0x157)			
20	0x24			

12.6.4.13.2.1 Disable IP/UDP/TCP Checksum Offloading 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	0x24			

12.6.4.14 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 12.4.6.

Note: All indexes in this command set start at one to match the NC-SI methodology.

12.6.4.14.1 Set IP Address Command (Intel Command 0x25, Parameter 0x0)

The Set IP Address command is used by the MC to communicate its IP Address to a NC.

If at least one IP Address filter is enabled, only unicast packets that match one of the enabled filters are forwarded through the NC-SI interface. Otherwise, the IP Address is ignored in the unicast filtering process.

This command does not impact the forwarding results. It is used as a preliminary stage to the Set Binding 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...23	0x25	0x0	Reserved	
24...27	Management Controller IP Address			
28...31				
32...35				
36...39				
40...43	Reserved		IP Address Number	Set IP Flags
44...47	Checksum			

Where:

- ***Management Controller IP Address*** — An IP Address that is used by the MC.
  - — If the *IP Version* bit of the *Set IP Flags* field is 0b (IPv4), this is a 4-byte unicast IPv4 Address in network byte order. In this case, the address occupies bytes 24-27 of the packet, and bytes 28-39 are ignored.
  - — If the *IP Version* bit of the *Set IP Flags* field is 1b (IPv6), this is a 16-byte unicast IPv6 Address in network byte order. In this case, the address occupies the full field (bytes 24-39 of the packet).
- ***IP Address Number*** — Indicates which IP Address filter is configured by the command. The value can relate to one of three pools of filters according to Table 12-23:

Table 12-23. IP Filter Pools

Set IP Flag.IP Version	Set IP Flag.Mixed Index	Pool to Use	Allowed Values
0	0	IPv4	1 to the number of IPv4 only addresses.
1	0	IPv6	1 to the number of IPv6 only addresses.
X (0/1)	1	Mixed	1 to the number of mixed IP Addresses.

Note: The values shown in the allowed values column refer to the Get Shared IP Capabilities Response (Section 12.6.4.14.6.1).

- ***Set IP Flags*** — The bits in this field are listed in Table 12-24:

Table 12-24. Set IP Flag Field

Bits	Field	Description
0	Enable	0b = Disable the filter. 1b = Enable the filter.
1	IP Version	0b = IPv4 1b = IPv6

Table 12-24. Set IP Flag Field [continued]

Bits	Field	Description
2	Mixed Index	0b = Index relates to the IPv4 or IPv6 only IP filter sets according to the *IP Version* field. 1b = Index relates to the mixed IP filter set.
3	MAC-based IP	This flags define if the IPv6 Address is derived from a MAC Address and thus only the 24 LSB should be used for the comparison. This flag is relevant only if the *IP Version* = IPv6 (1b). 0b = Filter according to the full 128 bits of IPv6 Address. 1b = Filter according to the 24 LS bits of the IPv6 Address.
7:4	Reserved	Reserved.

12.6.4.14.1.1 Set IP Address Response

The NC must, in the absence of a checksum error or identifier mismatch, always accept the Set IP Address command and send a response using 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	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x25	0x0	Reserved	
28...31	Checksum			

12.6.4.14.2 Get IP Address Command (Intel Command 0x25, Parameter 0x1)

An MC uses the Get IP Address command to determine the IP Address programmed in one of the IP Address filters in a NC.

	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	0x25	0x1	Reserved	
24...27	Reserved		IP Address Number	IP Filter Pool
28...31	Checksum			

Where:

- ***IP Address Number*** — Defines the index of the IP Address in the pool defined by the IP filter pool. The allowed values are listed in Table 12-23.
- ***IP Filter Pool***:

  0x0 = Mixed IP filters
  0x1 = IPv4 filters
  0x2 = IPv6 filters
  0x3-0xFF = Reserved

12.6.4.14.2.1 Get IP Address Response

The NC, in the absence of a checksum error or identifier mismatch, must always accept the Get IP Address command and send a response using 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	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x25	0x1	IP Address Number	Get IP Flags
28...31	Management Controller IP Address			
32...35				
36...39				
40...43				
44...47	Checksum			

Where:

- ***Management Controller IP Address*** — An IP Address that is used by the MC.
  - If the *IP Version* bit of the *Get IP Flags* field is 0b (IPv4), this is a 4-byte unicast IPv4 Address in network byte order. In this case, the address occupies bytes 28-31 of the packet, and bytes 32-43 are ignored.
  - If the *IP Version* bit of the Flags field is 1b (IPv6), this is a 16-byte unicast IPv6 Address in network byte order. In this case, the address occupies the full field (bytes 28-43 of the packet).
- ***IP Address Number*** — Indicates which IP Address filter is described in the response. Should be equal to the *IP Address Number* in the command.
- ***Get IP Flags*** — The bits in this field are listed in Table 12-25:

Table 12-25. Get IP Flag Field

Bits	Field	Description
0	Enable	0b = Filter is disabled. 1b = Filter is enabled.
1	IP Version	0b = IPv4 1b = IPv6
2	Mixed Index	0b = Index relates to the IPv4 or IPv6 only IP filter sets according to the *IP Version* field. 1b = Index relates to the mixed IP filter set.
3	MAC-based IP	This flags define if the IPv6 Address is derived from a MAC Address and thus only the 24 LSB should be used for the comparison. This flag is relevant only if the *IP Version* = IPv6 (1b). 0b = Filter according to the full 128 bits of IPv6 Address. 1b = Filter according to the 24 LS bits of the IPv6 Address.
7:4	Reserved	Reserved.

12.6.4.14.3 Set Port Command (Intel Command 0x25, Parameter 0x2)

An MC uses the Set Port command to communicate one of its TCP or UDP ports to a NC.

This command does not impact the forwarding results. It is used as a preliminary stage to the Set Binding command.

If the *Ignore Protocol* flag is cleared, the protocol should also match the *Protocol* field. Otherwise, the *Protocol* field is ignored.

	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	0x25	0x2	Set Port Flags	Reserved
24...27	Port Index	Protocol	Port	
28...31	Checksum			

Where:

- ***Protocol*** — The value to match in the IPv4 header *Protocol* field or IPv6 header *Next Header* field. These values are defined by IANA. Allowed values are 0x6 (TCP) and 0x11 (UDP).
- ***Port*** — The value to match in the *Destination Port* or *Source Port* field of the TCP or UDP header. The legal port range for both TCP and UDP is 0-65,535. The compared field is defined by the *Port Type* flag.
- ***Port Index*** — Indicates which port filter is configured by the command. Allowed values are 1 to n, where n is the number of port filters supported by the Network Controller.
- ***Set Port Flags*** — The bits in this field are listed in Table 12-26.

Table 12-26. Set Port Flags Field Descriptions

Bits	Field	Description
0	Enable	0b = Disable the filter. 1b = Enable the filter.
1	Ignore Protocol	0b = Filter by port and protocol. 1b = Filter by port only.
2	Port Type	0b = Compare destination port. 1b = Compare source port.
7:3	Reserved	Reserved.

12.6.4.14.3.1 Set Port Response

The NC must, in the absence of a checksum error or identifier mismatch, always accept the Set Port command and send a response using 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	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x25	0x2	Reserved	
28...31	Checksum			

12.6.4.14.4 Get Port Command (Intel Command 0x25, Parameter 0x3)

An MC uses the Get Port command to determine the TCP or UDP port programmed in one of the port filters in a NC.

	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	0x25	0x3	Reserved	
24...27	Reserved		Port Index	Reserved
28...31	Checksum			

Table 12-27 lists the fields in the Get Port command.

Table 12-27. Get Port Command Field Descriptions

Field	Field Description	Value Description
Port Index	Indicates which port filter is requested by the command.	1 to n, where n is the number of port filters supported by the NC.

12.6.4.14.4.1 Get Port Response

The NC must, in the absence of a checksum error or identifier mismatch, always accept the Get Port command and send a response using 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	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x25	0x3	Get Port Flags	Reserved
28...31	Port Index	Protocol	Port	
32...35	Checksum			

Where:

- ***Protocol*** — The value compared in the IPv4 header *Protocol* field or IPv6 header *Next Header* field. Possible values are 0x6 (TCP) and 0x11 (UDP).
- ***Port*** — The value compared in the *Destination Port* or *Source Port* field of the TCP or UDP header.
- ***Port Index*** — Indicates which port filter is reported by the response. Should match the *Port Index* in the command.
- ***Get Port Flags*** – The bits in this field are listed in Table 12-28.

Table 12-28. Get Port Flags Field Descriptions

Bits	Field	Description
0	Enable	0b = Filter is disabled. 1b = Filter is enabled.
1	Ignore Protocol	0b = Filter by port and protocol. 1b = Filter by port only.
2	Port Type	0b = Compare destination port. 1b = Compare source port.
7:3	Reserved	Reserved.

12.6.4.14.5 Enable Unicast Infrastructure Filter Command (Intel Command 0x25, Parameter 0x4)

A MC uses the Enable Unicast Infrastructure Filter command to configure a NC to forward copies of network infrastructure packets to it.

Network infrastructure packets contain messages that are necessary for operating the network infrastructure layers (such as DHCP, ARP, and DNS messages). This is required when the MC shares an IP Address with the host. In this case, both the host and the MC need to process the messages. As a result, the NC must forward the packets to both the MC and the host.

This command should be applied only after a MAC Address is added using the Set MAC Address NC-SI command.

All the IP Addresses added through the Set IP command before this command is given are considered as IP Addresses of the MC for the purpose of this command.

If a Set IP command is received after this command was received, the list of IP Address is not updated, and this command should be given again.

	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	0x25	0x4	Reserved	
28...31	Unicast Infrastructure Filter Settings			
32...35	Checksum			
36...39	Padding			

Table 12-29 lists the sub fields of the *Unicast Infrastructure Filter Settings* field.

Table 12-29. Unicast Infrastructure Packet Filter Settings Field

Bits	Field	Description
0	ARP Response Packets Received From Wire	**ARP Response Packets Received From Wire** 0b = Forward this packet type to the host only 1b = Forward this packet type to both the host and the MC. For the purposes of this filter, an ARP response packet is defined to be any packet that meets all of the following requirements: • The *Ethernet Destination Address* field contains the MAC Address assigned to the MC. • The *Ethernet Type* field contains 0x0806 (ARP). • The *ARP Opcode* field is set to 0x0002 (response). • The A*RP Target Protocol Address* field contains the IP Address assigned to the MC.

Table 12-29. Unicast Infrastructure Packet Filter Settings Field [continued]

Bits	Field	Description
1	ICMPv4 Request Packets Received From Wire	**ICMPv4 Request Packets Received From Wire** 0b = Forward this packet type to the host only. 1b = Forward this packet type to both the host and the MC. For the purposes of this filter, an ICMP request packet is defined to be any packet that meets all of the following requirements: • The *Ethernet Destination Address* field contains the MAC Address assigned to the MC. • The *Ethernet Type* field contains 0x0800 (IPv4). • The *IP Destination Address* field contains the IPv4 Address assigned to the MC. • The *IP Protocol* field contains 1 (ICMP).
2	ICMPv6 Request Packets Received From Wire	**ICMPv6 Request Packets Received From Wire** 0b = Forward this packet type to the host only. 1b = Forward this packet type to both the host and the MC. For the purposes of this filter, an ICMPv6 request packet is defined to be any packet that meets all of the following requirements: • The *Ethernet Destination Address* field contains the MAC Address assigned to the MC. • The *Ethernet Type* field contains 0x86DD (IPv6). • The *IP Destination Address* field contains the IPv6 Address assigned to the MC. • The *IP Next Header* field contains 58 (ICMPv6). ***Note:*** This filter is not supported by the E810.
3	DHCP Server Unicast Packets Received From Wire	**DHCP Server Unicast Packets Received From Wire** 0b = Forward this packet type to the host only. 1b = Forward this packet type to both the host and the MC. For the purposes of this filter, a DHCP server unicast packet is defined to be any packet that meets all of the following requirements: • The *Ethernet Destination Address* field contains the MAC Address assigned to the MC. • The *Ethernet Type* field contains 0x0800 (IPv4). • The *IP Destination Address* field contains either 255.255.255.255 (the local broadcast address) or the IPv4 Address assigned to the MC. • The *IP Protocol* field contains 17 (UDP). • The *UDP Destination Port* field contains 68 (bootstrap protocol client).
4	DNS Server Packets Received From Wire	**DNS Server Packets Received From Wire** 0b = Forward this packet type to the host only. 1b = Forward this packet type to both the host and the MC. For the purposes of this filter, a DNS server unicast packet is defined to be any packet that meets all of the following requirements: • The *Ethernet Destination Address* field contains the MAC Address assigned to the MC. • The *Ethernet Type* field contains 0x0800 (IPv4). • The *IP Destination Address* field contains the IPv4 Address assigned to the MC. • The *IP Protocol* field contains 17 (UDP). • The *UDP Source Port* field contains 53 (domain name server).
5	DHCP Client Packets Transmitted By Host	**DHCP Client Packets Transmitted By Host** 0b = Forward this packet type to the wire only. 1b = Forward this packet type to both the wire and the MC. For the purposes of this filter, a DHCP client unicast packet is defined to be any packet that meets all of the following requirements: • The *Ethernet Source Address* field contains the MAC Address assigned to the MC. • The *Ethernet Type* field contains 0x0800 (IPv4). • The *IP Protocol* field contains 17 (UDP). • The *UDP Destination Port* field contains 67 (bootstrap protocol server).

Table 12-29. Unicast Infrastructure Packet Filter Settings Field [continued]

Bits	Field	Description
6	DHCPv6 Server Unicast Packets Received From Wire	**DHCPv6 Server Unicast Packets Received From Wire** 0b = Forward this packet type to the host only. 1b = Forward this packet type to both the host and MC. For the purposes of this filter, a DHCPv6 server unicast packet is defined to be any packet that meets all of the following requirements: • The *Ethernet Destination Address* field contains the MAC Address assigned to the MC. • The *Ethernet Type* field contains 0x86DD (IPv6). • The *IPv6 Destination Address* field contains the IPv6 Address assigned to the MC. • The *IP Protocol* field contains 17 (UDP). • The *UDP Destination Port* field contains 546 (DHCPv6 protocol client).
7	RMCP Primary Port - UDP	**RMCP Primary Port - UDP** 0b = Forward this packet type to the host. 1b = Forward this packet type to the MC only. For the purposes of this filter, a RMCP primary UDP packet is defined to be any packet that meets all of the following requirements: • The *Ethernet Destination Address* field contains the MAC Address assigned to the MC. • The *Ethernet Type* field contains 0x86DD (IPv6) 0r 0x0800 (IPv4). • The *IP Destination Address* field contains the one of the IP Addresses assigned to the MC. • The *IP Protocol* field contains 17 (UDP). • The *UDP Destination Port* field contains 623 [aux bus shunt (primary RMCP port)].
8	RMCP Primary Port - TCP	**RMCP Primary Port - TCP** 0b = Forward this packet type to the host. 1b = Forward this packet type to the MC only. For the purposes of this filter, a RMCP primary TCP packet is defined to be any packet that meets all of the following requirements: • The *Ethernet Destination Address* field contains the MAC Address assigned to the MC. • The *Ethernet Type* field contains 0x86DD (IPv6) 0r 0x0800 (IPv4). • The *IP Destination Address* field contains the one of the IP Addresses assigned to the MC. • The *IP Protocol* field contains 6 (TCP). • The *UDP Destination Port* field contains 623 [aux bus shunt (primary RMCP port)].
9	RMCP Secondary Port - UDP	**RMCP Secondary Port - UDP** 0b = Forward this packet type to the host 1b = Forward this packet type to the MC only. For the purposes of this filter, a RMCP secondary UDP packet is defined to be any packet that meets all of the following requirements: • The *Ethernet Destination Address* field contains the MAC Address assigned to the MC. • The *Ethernet Type* field contains 0x86DD (IPv6) 0r 0x0800 (IPv4). • The *IP Destination Address* field contains the one of the IP Addresses assigned to the MC. • The *IP Protocol* field contains 17 (UDP). • The *UDP Destination Port* field contains 664 [secure aux bus (secondary RMCP port)].
10	RMCP Secondary Port - TCP	**RMCP Secondary Port - TCP** 0b = Forward this packet type to the host. 1b = Forward this packet type to the MC only. For the purposes of this filter, a RMCP secondary TCP packet is defined to be any packet that meets all of the following requirements: • The *Ethernet Destination Address* field contains the MAC Address assigned to the MC. • The *Ethernet Type* field contains 0x86DD (IPv6) 0r 0x0800 (IPv4). • The *IP Destination Address* field contains the one of the IP Addresses assigned to the MC. • The *IP Protocol* field contains 6 (TCP). • The *UDP Destination Port* field contains 664 [secure aux bus (secondary RMCP port)].
31:11	Reserved	Reserved.

12.6.4.14.5.1 Enable Unicast Infrastructure Filter Response

The NC, in the absence of a checksum error or identifier mismatch, must always accept the Enable Unicast Infrastructure Filter command and send a response using the following format.

Note: Currently, no command-specific reason codes are identified for this 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	0x25	0x4	Reserved	
28...31	Checksum			

12.6.4.14.6 Get Shared IP Capabilities Command (Intel Command 0x25, Parameter 0x5)

An MC uses the Get Shared IP Capabilities command to determine the level of support of shared IP of the device.

	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	0x25	0x5	Reserved	
24...27	Checksum			

12.6.4.14.6.1 Get Shared IP Capabilities Response

The NC must, in the absence of a checksum error or identifier mismatch, always accept the Get Shared IP Capabilities command, and send a response using the following format.

Note: Currently, no command-specific reason codes are identified for this 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	0x25	0x5	Number of Mixed IP Addresses	Number of IPv4-only Addresses
28...31	Number of IPv6-only Addresses	Number of Ports	Numbers of Bindings	Filtering Capabilities
32...35	Unicast Infrastructure Filter Capabilities			
36...39	Checksum			

Where:

- ***Number of Mixed IP Addresses*** — The number of supported IP filters that can be used for IPv4 or IPv6. The E810 does not support mixed IP Address filters.
- ***Number of IPv4-only Addresses*** — The number of supported IP filters that can be used for IPv4 only. The E810 supports three IPv4 Address filters.
- ***Number of IPv6-only Addresses*** — The number of supported IP filters that can be used for IPv6 only. The E810 supports four IPv6 Address filters.
- ***Number of Ports*** — The number of supported port filters.
- ***Number of Bindings*** — Defines the number of IP Addresses that can be bound with different ports.
- ***Unicast Infrastructure Filter Capabilities*** — Defines the optional unicast infrastructure filter capabilities that the channel supports. The bit definitions for this field correspond directly with the bit definitions for the Unicast Infrastructure Filter Settings field defined for the Unicast Infrastructure Filter command listed in Table 12-29. A bit set to 1b indicates that the channel supports the filter associated with that bit position. Otherwise, the channel does not support that filter. The E810 supports all filters but ICMPv6 filtering, so the returned value is 0x7FB.
- ***Filtering Capabilities —*** The bits in this field are listed in Table 12-30.

Table 12-30. Filtering Capabilities Field Descriptions

Bits	Field	Description
0	IPv4 Support	0b = IPv4 filtering is not supported. 1b = IPv4 filtering is supported.
1	IPv6 Support	0b = IPv6 filtering is not supported. 1b = IPv6 filtering is supported.
2	Protocol Filtering Support	0b = Filtering by protocol is not supported. 1b = Filtering by protocol is supported.
3	Source Port Filtering Support	0b = Port filtering is supported only for destination port. 1b = Port filtering is supported for destination port or source port.
7:4	Reserved	Reserved.

12.6.4.14.7 Shared IP Enable Broadcast Filtering Command (Intel Command 0x25, Parameter 0x6)

The Shared IP Enable Broadcast Filtering command is defined to enable the MC to limit the flow of ARP requests to those that contain a target IP Address value that matches the MC IP Address.

This command should be used instead of the regular NC-SI Enable Broadcast Filtering command.

Note: Receiving a standard NC-SI Enable Broadcast Filtering command enables the matching bits in this command. Receiving a standard NC-SI Disable Broadcast Filter Command clears the settings in 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	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x25	0x6	Reserved	
28...31	Shared IP Broadcast Packet Filter Settings			
32...35	Checksum			
36...39	Padding			

The content of the *Shared IP Broadcast Packet Filter Settings* field is listed in Table 12-31. Bit 4 has been added to the standard Enable Broadcast Filtering command limit ARP broadcast packets to the MC IP Address.

Table 12-31. Shared IP Broadcast Packet Filter Settings Field

Bits	Field Description	Value Description
3:0	As defined in DSP0222.	As defined in DSP0222 in 8.4.33 Enable Broadcast Filter Command (0x10) - Table 68.
4	Limit ARP Broadcast Packets to Management Controller IP Address.	When Bit 0 is set, it limits the flow of ARP packets to the MC as follows: 0b = Forward all ARP broadcast packets to the MC. 1b = Forward only ARP broadcast packets that are targeted at IP Addresses bound to the MC. All the IPs set by the Set IP command before this command is given are included in forwarding. This field is optional. If unsupported, the behavior for ARP packets is set according to Bit 1 in this structure. The value must be set to 0b if unsupported.
31:5	Reserved	Reserved.

12.6.4.14.7.1 Shared IP Enable Broadcast Filtering Response

The NC must, in the absence of a checksum error or identifier mismatch, always accept the Shared IP Enable Broadcast Filtering command, and send a response using the following format.

Note: Currently, no command-specific reason codes are identified for this 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	0x25	0x6	Reserved	
28...31	Checksum			

12.6.4.14.8 Shared IP Enable Global Multicast Filtering Command (Intel Command 0x25, Parameter 0x7)

The Shared IP Enable Global Multicast Filtering command is defined to enable the MC to enable the forwarding of IEEE 802.1X Extensible Authentication Protocol over LAN (EAPOL) frames to the MC IP Address. IEEE 802.1X defines methods for port-based network access control.

This command should be used instead of the regular NC-SI Enable Global Multicast Filtering command.

Note: Receiving a standard NC-SI Enable Global Multicast Filtering command enables the matching bits in this command. Receiving a standard NC-SI Disable Global Multicast Filter command clears the settings in 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	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x25	0x7	Reserved	
28...31	Shared IP Multicast Packet Filter Settings			
32...35	Checksum			
36...39	Padding			

The content of the *Shared IP Multicast Packet Filter Settings* field is listed in Table 12-32. Bit 3 has been added to the standard Enable Broadcast Filtering command limit ARP broadcast packets to MC IP Address.

Table 12-32. Shared IP Multicast Packet Filter Settings Field

Bits	Field Description	Value Description
2:0	As defined in DSP0222.	As defined in DSP0222 in 8.4.37 Enable Global Multicast Filter Command (0x12) - Table 74.
3	IEEE 802.1X EAPOL	This field is optional. If unsupported, multicast 802.1X packets are blocked when multicast filtering is enabled, unless they are matched by an address filter configured using the Set MAC Address command. The value must be set to 0b if unsupported. 0b = Filter out this packet type. 1b = Forward this packet type to the MC. For the purposes of this filter, a IEEE 802.1X multicast packet is defined to be any packet that meets all of the following requirements: • The *Destination MAC Address* field is set to the layer 2 multicast address 01:80:c2:00:00:03. • The *EtherType* field is set to 0x888E (802.1X PAE).
31:4	Reserved	Reserved.

12.6.4.14.8.1 Shared IP Enable Global Multicast Filtering Response

The NC must, in the absence of a checksum error or identifier mismatch, always accept the Shared IP Enable Global Multicast Filtering command, and send a response using the following format.

Note: Currently, no command-specific reason codes are identified for this 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	0x25	0x7	Reserved	
28...31	Checksum			

12.6.4.14.9 Get Shared IP Parameters Command (Intel Command 0x25, Parameter 0x8)

The Get Shared IP Parameters command can be used by the MC to request that the channel send the MC a copy of part of the currently-stored parameter settings that have been put into effect by the MC related to shared IP filtering.

	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	0x25	0x8	Reserved	
24...27	Checksum			

12.6.4.14.9.1 Get Shared IP Parameters Response

The NC must, in the absence of a checksum error or identifier mismatch, always accept the Get Shared IP Parameters command, and send a response using the following format.

Note: Currently, no command-specific reason codes are identified for this 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	0x25	0x8	Reserved	
28...31	Number of IP Addresses	IP Address Flags		
32...35	Number of Ports	Port Flags		
36...39	Unicast Infrastructure Filter Settings			
40...43	Broadcast Filtering Settings			
44...47	Multicast Filtering Settings			
48...51	Checksum			

Where:

- ***Number of IP Addresses*** — The number of supported IP filters including all the types of IP Addresses (IPv4 only, IPv6 only and mixed).
- ***IP Address Flags*** — The enable/disable state for each supported IP Address. See Table 12-33.

Table 12-33. IP Address Flags Field

Bits	Field Description	Value Description
0	IP Address 1 Status	0b = Default or unsupported or disabled. 1b = Enabled.
1	IP Address 2 Status or Reserved	0b = Default or unsupported or disabled. 1b = Enabled.
2	IP Address 3 Status or Reserved	0b = Default or unsupported or disabled. 1b = Enabled.
...		
23	IP Address 24 Status or Reserved	0b = Default or unsupported or disabled. 1b = Enabled.

Note: IP Address flags are organized in the following order: IPv4 Addresses first, followed by IPv6 Addresses, followed by mixed addresses, with the number of each corresponding to those reported through the Get Shared IP Capabilities command.

For example, if the interface reports four IPv4 filters, two IPv6 filters, and two mixed filters, IP Addresses 1 through 4 are those currently configured through the interface's IPv4 filters, IP Addresses 5 and 6 are those configured through the IPv6 filters, and 7 and 8 are those configured through the mixed filters.

The actual settings of each enabled IP Address can be found using the Get IP Address command

- ***Number of Ports*** — The number of supported port filters.
- ***Port Flags*** — The enable/disable state for each supported ports. See Table 12-34.

Table 12-34. Port Flags Field

Bits	Field Description	Value Description
0	Port 1 Status	0b = Default or unsupported or disabled. 1b = Enabled.
1	Port 2 Status or Reserved	0b = Default or unsupported or disabled. 1b = Enabled.
2	Port 3 Status or Reserved	0b = Default or unsupported or disabled. 1b = Enabled.
...		
23	Port 24 Status or Reserved	0b = Default or unsupported or disabled. 1b = Enabled.

Note: The actual settings of each enabled port can be found using the Get Port command.

- ***Unicast Infrastructure Filter Settings*** — Defines the optional unicast infrastructure filter capabilities settings. The bit definitions for this field correspond directly with the bit definitions for the *Unicast Infrastructure Filter Settings* field defined for the Unicast Infrastructure Filter command in Table 12-29. A bit set to 1b indicates that the filter associated with that bit position is enabled. Otherwise, the filter is not enabled.
- ***Broadcast Filter Settings*** — Defines the optional broadcast filter settings. The bit definitions for this field correspond directly with the bit definitions for the *Broadcast Filter Settings* field defined for the Shared IP Broadcast Filtering command in Table 12-31. A bit set to 1b indicates that the filter associated with that bit position is enabled. Otherwise, the filter is not enabled.
- ***Multicast Filter Settings*** — Defines the optional multicast filter capabilities settings. The bit definitions for this field correspond directly with the bit definitions for the *Multicast Filter Settings* field defined for the Shared IP Global Multicast Filtering command in Table 12-32. A bit set to 1b indicates that the filter associated with that bit position is enabled. Otherwise, the filter is not enabled.

12.6.4.14.10 Set Binding Command (Intel Command 0x25, Parameter 0x9)

The Set Binding command is used by the MC to define which combination of MAC Addresses, VLAN tags, IP Addresses and TCP/UDP ports should be forwarded to the MC.

	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	0x25	0x9	Binding Index	Set Binding Flags
24...27	Enabled MAC Addresses			
28...31	Enabled VLAN			
32...35	Enabled IP Addresses			
36...39	Enabled Ports (MSB)			
40...43	Enabled Ports (LSB)			
44...47	Checksum			

Once a Set Binding command is activated, all the previous forwarding rules based on the Set MAC Address or Set VLAN Filter commands are disabled and should be re-enabled using the Set Binding command. Subsequent Set MAC Address or Set VLAN Filter commands are used to enable MAC or VLAN addresses for the Set Binding command, but does not impact the forwarding rules.

Table 12-35 lists the fields in the Set Binding Flags field.

Table 12-35. Set Binding Flags Field Descriptions

Bits	Field Description	Value Description
0	Enable	0b = Disable the binding. 1b = Enable the binding.
1	Exclusive to MC	0b = Traffic matching this filter is sent to the MC and to the host. 1b = Traffic matching this filter is sent to the MC only.
2	Apply to Network[1]	0b = Do not compare traffic received from the network when checking this binding. 1b = Compare traffic received from the network when checking this binding.
3	Apply to Host	0b = Do not compare traffic received from the host when checking this binding. 1b = Compare traffic received from the host when checking this binding.
7:4	Reserved	Reserved.

1. At least one of the apply to network/host flags should be set for enabled bindings. Clearing both of them is equivalent to disabling the filter.

Where:

- ***Binding Index*** — Indicates which binding is configured by the command. The value should be smaller than the number of supported bindings as reported in the Get Shared IP Capabilities response in the *Number of Bindings* field.

- ***Enabled MAC Addresses*** — The MAC Addresses participating in this binding. The numbering of the MAC Addresses is similar to the one used in the *Mac Address Flags* in the Get Parameters response. Namely, MAC Addresses are returned in the following order: unicast filtered addresses first, followed by multicast filtered addresses, followed by mixed filtered addresses, with the number of each corresponding to those reported through the Get Capabilities command. A MAC Address can be added to a binding only if previously enabled through a Set MAC Address NC-SI command
- ***Enabled VLAN*** — The VLAN IDs participating in this binding. The numbering of the VLAN IDs. A VLAN tag can be added to a binding only if previously enabled through a Set VLAN Filter NC-SI command.
- ***Enabled IP Addresses*** — The IP Addresses participating in this binding. The numbering of the IP Addresses is similar to the one used in Section 12.6.4.14.9. An IP Address can be added to a binding only if previously enabled through a Set IP Address Intel OEM command.
- ***Enabled Ports*** — The ports participating in this binding. A port can be added to a binding only if previously enabled through a Set Port Intel OEM command.

12.6.4.14.10.1 Set Binding Response

The NC must, in the absence of a checksum error or identifier mismatch, always accept the Set Binding command and send a response using 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	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x25	0x9	Reserved	
28...31	Checksum			

12.6.4.14.11 Get Binding Command (Intel Command 0x25, Parameter 0xA)

The Get Binding command to is used by the MC to determine the current programming of one of the bindings in a NC.

	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	0x25	0xA	Binding Number	Reserved
24...27	Checksum			

Where:

- ***Binding Number*** — Indicates which binding is requested by the command. The value should be smaller than the number of supported bindings as reported in the Get Shared IP Capabilities Response in the *Number of Bindings* field.

12.6.4.14.11.1 Get Binding Response

The NC must, in the absence of a checksum error or identifier mismatch, always accept the Get Binding command, and send a response using 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	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x25	0xA	Binding Number	Get Binding Flags
28...31	Enabled MAC Addresses			
32...35	Enabled VLAN			
36...39	Enabled IP Addresses			
40...43	Enabled Ports (MSB)			
44...47	Enabled Ports (LSB)			
48...51	Checksum			

The fields in the Get Binding response are equivalent to their counterparts in the Set Binding command.

12.6.4.14.12 Set Shared Mode Command (Intel Command 0x25, Parameter 0xB)

An MC uses the Set Shared Mode command to indicate to the NIC that it intends to operate in shared MAC/ IP mode or in dedicated MAC mode.

If used, this command should be sent before any of the regular or OEM NC-SI commands used to set forwarding filters. When this command is received, all the filters are cleared.

This command is needed only when the Intel OEM commands with command ID 0x25 are used to configure the shared behavior. If other commands are used, users should ensure the correct configuration of the filters.

When shared mode is activated, the Set MAC and Set VLAN NC-SI commands do not impact the receive filtering until a Set Binding or Enable Unicast Infrastructure Filter command is received.

Any other command from this section (Section 12.6.4.14) received before shared mode is set fails with a "Not in Shared Mode" (0x5092) reason.

	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	0x25	0xB	Shared Mode	Reserved
24...27	Checksum			

Where:

- ***Shared Mode***:

  0x0: Dedicated MAC mode.

  0x1: Shared MAC/IP mode.

12.6.4.14.12.1 Set Shared Mode Response

The NC must, in the absence of a checksum error or identifier mismatch, always accept the Set Shared Mode command, and send a response using 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	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x25	0xB	Shared Mode	Reserved
28...31	Checksum			

12.6.4.15 OS2BMC Configuration

These commands control enabling of the OS2BMC flow.

12.6.4.15.1 Enable OS2BMC Flow Command (Intel Command 0x40, Parameter 0x01)

	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	0x01		

12.6.4.15.1.1 Enable OS2BMC Flow 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	0x40	0x01		

12.6.4.15.2 Enable Network-to-BMC Flow Command (Intel Command 0x40, Parameter 0x02)

	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		

12.6.4.15.2.1 Enable Network-to-BMC Flow 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	0x40	0x02		

12.6.4.15.3 Enable Both Enable Both Network-to-BMC and Host-to-BMC Flows Command (Intel Command 0x40, Parameter 0x03)

	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		

12.6.4.15.3.1 Enable Both Network-to-BMC and Host-to-BMC Flows 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	0x40	0x03		

12.6.4.15.4 Set BMC IP Address Command (Intel Command 0x40, Parameter 0x04)

This command is used to expose the MC IP Address to the host. This command is supported by the E810, 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 indicates whether the IP Address is an IPv4 or an IPv6 Address:

0 = IPv4

1 = IPv6

2 = No IP Address — The command should not include an IP Address.

	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	0x40	0x04	IP Type	IPv6 Address (MSB, byte 15)/IPv4 Address (MSB, byte 3)
24...27	IPv6 Address (byte 14)/ IPv4 Address (byte 2)	IPv6 Address (byte 13)/ IPv4 Address (byte 1)	IPv6 Address (byte 12)/ IPv4 Address (LSB, byte 0)	IPv6 Address (byte 11)/ Reserved
28...31	...	...	...	...
32...35	...	...	...	...
36...38	...	...	IPv6 Address (LSB, byte 0)/Reserved	

12.6.4.15.4.1 Set BMC IP Address 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	0x40	0x04		

12.6.4.15.5 Get OS2BMC Parameters Command (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	Manufacturer ID (Intel 0x157)			
20	0x41			

12.6.4.15.5.1 Get OS2BMC Parameters 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	0x41	Status	IPv6 Address (MSB, byte 15)/IPv4 Address (MSB, byte 3)	IPv6 Address (byte 14)/ IPv4 Address (byte 2)
28...31	IPv6 Address (byte 13)/ IPv4 Address (byte 1)	IPv6 Address (byte 12)/ IPv4 Address (LSB, byte 0)	IPv6 Address (byte 11)/ Reserved	...
32...35	...	...	...	...
36...39	...	...	...	...
40...41	...	IPv6 Address (LSB, byte 0)/Reserved		

Where the *Status* byte partition is as follows:

Table 12-36. Status Byte Description

Bits	Content
0	Relevant only if the I*P Address Valid* bit is set. 0b = IPv4 1b = IPv6
1	IP Address Valid Never valid for the E810.
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.

12.6.4.16 Diagnostic Commands

12.6.4.16.1 Get Controller Information Command (Intel Command 0x48, Parameter 0x1)

This command gathers the controller identification information and return it back to the MC.

	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	0x48	0x1		

12.6.4.16.1.1 Get Controller Information 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	0x48	0x1	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 12-37. 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	
0x16	2	FCoE Boot Firmware Version	

12.6.4.17 ASIC Temperature Value

12.6.4.17.1 Get ASIC Temperature Command (Intel Command 0x4B)

This is a per spec implementation of NC-SI 1.2 Get ASIC Temperature command.

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header (0x50)			
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			

12.6.4.17.1.1 Get ASIC Temperature Response

	Bits			
Bytes	**31:24**	**23:16**	**15:8**	**7:0**
0...3	NC-SI Header (0xD0)			
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			

Where:

- ***Maximum Temperature Value*** — This value is the maximum T-Diode temperature limit in °C at which the E810 can operate at full load for its rated service lifetime. The value should be de-rated to take measurement tolerance into account. The value is reported as a hexadecimal integer number. The value to report for the E810 is: 102 - 0x66 (105 - Thermal sensor accuracy of +/-3).
- ***Current Temperature Value*** — This value is the current real-time temperature of the chip in °C. The value is reported as a hexadecimal integer number.

12.6.4.18 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 can 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				
20...23	Reserved			0x82
24	Index of firmware module in which the error was found. The encoding is as defined in GL_MNG_FWSM.*EXT_ERR_IND* field.			

12.6.5 Basic NC-SI Workflows

12.6.5.1 Package States

A NC package can be in one of the following two states:

- **Selected** — The package is allowed to use the NC-SI lines, meaning the NC package might send data to the MC.
- **Deselected** — 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:

- **Select Package command** — This command explicitly selects the NC package.
- 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 Deselect 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.

12.6.5.2 Channel States

A NC channel can be in one of the following states:

- **Initial State** — The channel only accepts the Clear Initial State command (the package also accepts the Select Package and Deselect Package commands).
- **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.

12.6.5.3 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:

For *package_id* = 0x0 to MAX_PACKAGE_ID:

1. Issue a Select Package command to package ID *package_id*.

   If a response was received:

   a. For *internal_channel_id* = 0x0 to MAX_INTERNAL_CHANNEL_ID.

   b. 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).

      If a response was received:

      1. Consider *internal_channel_id* as a valid channel for the *package_id* package.
      2. The MC can now optionally discover channel capabilities and version ID for the channel.

      Else, if a response was not received:

      1. Issue a Clear Initial State command three times.

   c. Issue a Deselect Package command to the package (and continue to the next package).

   Else, if a response was not received:

   a. Issue a Select Packet command three times.

12.6.5.4 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 reconfigure everything at power-up and channel/package resets.

12.6.5.4.1 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.

12.6.5.4.2 Receive Filtering

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.

12.6.5.4.2.1 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 E810 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.

For further details, refer to the Set MAC Address command in the NC-SI specification.

Broadcast (Enable/Disable Broadcast Filter command)

NC-SI defines a broadcast filtering mechanism that has the following states:

- **Enabled** — All broadcast traffic is blocked (not forwarded) to the BMC, except for specific filters (such as ARP request, DHCP, and NetBIOS).
- **Disabled** — All broadcast traffic is forwarded to the BMC, with no exceptions.

For further details, refer to the Enable/Disable Broadcast Filter command in the NC-SI specification.

Global Multicast (Enable/Disable Global Multicast Filter command)

NC-SI defines a multicast filtering mechanism which has the following states:

- **Enabled** — All multicast traffic is blocked (not forwarded) to the BMC.
- **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. For further details, refer to the Enable/Disable Global Multicast Filter command in the NC-SI specification.

12.6.5.4.3 VLAN

NC-SI defines the following VLAN work modes:

Mode	Command and Name	Description
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.

For further details, refer to the Enable VLAN command in the NC-SI specification.

The E810 only supports modes #1 and #3. Recommendation:

1. Modes:
   - If VLAN is not required, use the disabled mode.
   - 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.

12.6.5.5 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.

12.6.5.6 Channel Enable

This mode controls the state of the receive path:

- **Disabled** — The channel does not pass any traffic from the network to the MC.
- **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.

12.6.5.7 Network Transmit Enable

This mode controls the state of the transmit path:

- **Disabled** — The channel does not pass any traffic from the MC to the network.
- **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 12.6.9.3.

12.6.6 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, and so on).

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.

12.6.7 Querying Active Parameters

The MC can use the Get Parameters command to query the current status of the operational parameters.

12.6.8 Resets

In NC-SI there are two types of resets defined:

- Synchronous entry into the initial state.
- 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 reconfigure 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, and so on) 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 reconfigure 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.

12.6.9 Advanced Workflows

12.6.9.1 Multi-NC Arbitration

As described in Section 12.6.1.2, in a multi-NC environment, there is a need to arbitrate the NC-SI lines. Figure 12-9 shows the system topology of such an environment.

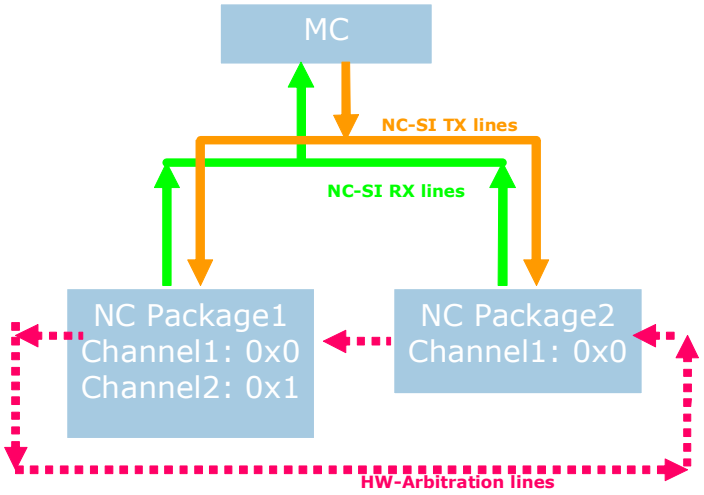


Figure 12-9. Multi-NC Environment

In Figure 12-9, 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 deselected.

NC-SI has defined two mechanisms for the arbitration scheme:

- **Package selection by the MC** — In this mechanism, the MC is responsible for arbitrating between the packages by issuing NC-SI commands (Select/Deselect Package). The MC is responsible for having only one package selected at any given time.
- **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.

12.6.9.2 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 Deselect 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 Deselect Package command to the *active_package*. The MC must then wait for a response and then an additional timeout for the package to become deselected (200 μs). 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*.

12.6.9.3 Multiple Channels (Fail-Over)

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:

- 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.
- Link Status Change AEN or Get Link Status command.

12.6.9.3.1 Fail-Over Algorithm Example

The following is a sample workflow for a fail-over scenario for the E810 (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).

      If found:

      1. Enable network Tx for that specific channel.
      2. 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.
      3. Resume normal workflow.

      If not found:

      1. 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.

12.6.9.4 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:

- **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.
- **NC-SI Statistics** — These are statistics on the NC-SI control frames (such as commands, responses, AENs, and so on). See the NC-SI specification for details.
- **NC-SI PT Statistics** — These are statistics on the NC-SI PT frames. See the NC-SI specification for details.

12.7 Management Component Transport Protocol (MCTP)

12.7.1 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 exists 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.

12.7.1.1 MCTP Usage Model

The E810 supports MCTP protocol over the PCIe and SMBus buses. The E810 can connect through MCTP to a MC or the ME engine in the chipset, as described in Figure 12-10.

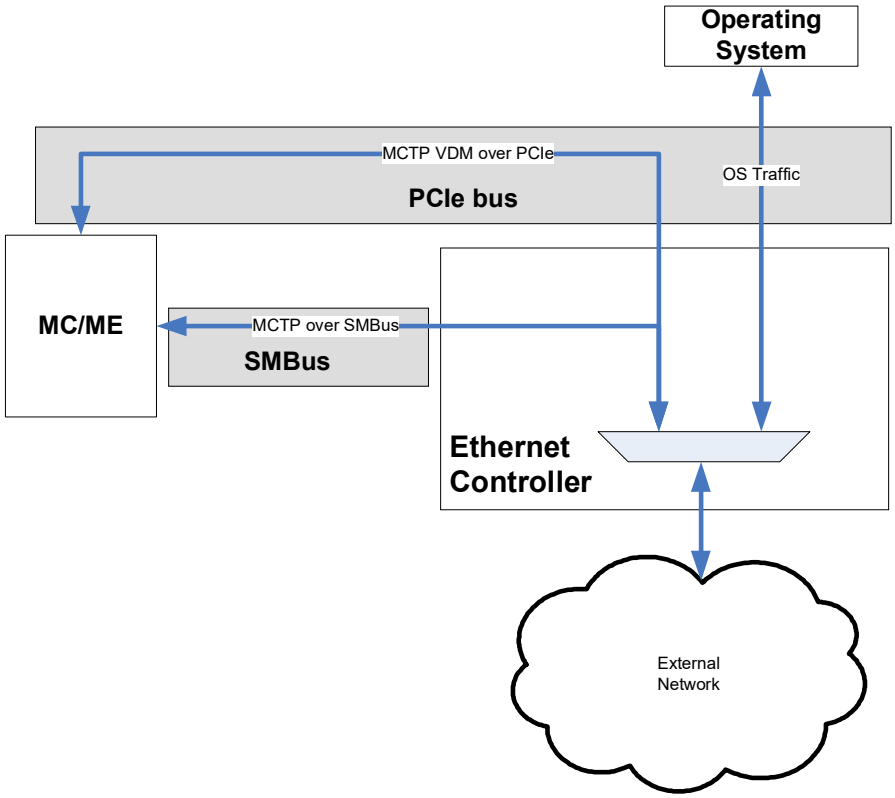


Figure 12-10. E810 MCTP Connections

12.7.1.2 Detecting an MC EID and Physical Address

To enable transactions between the MC and the NIC, the bus physical address (SMBus or PCIe) and the EID of the partner need to be discovered. NICs do not try to discover the MC, and assume the MC initiates the connection. If the NIC is in an NC-SI initial state, 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 E810. Subsequent pass-through 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.

12.7.1.3 Bus Transition

The following section defines the transition flow between PCIe and SMBus as the bus on which MCTP flows. Figure 12-11 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 of these variables are initialized to zero at power-on, apart from the SMBus address of the endpoint (NIC), which might be initialized from an NVM value.

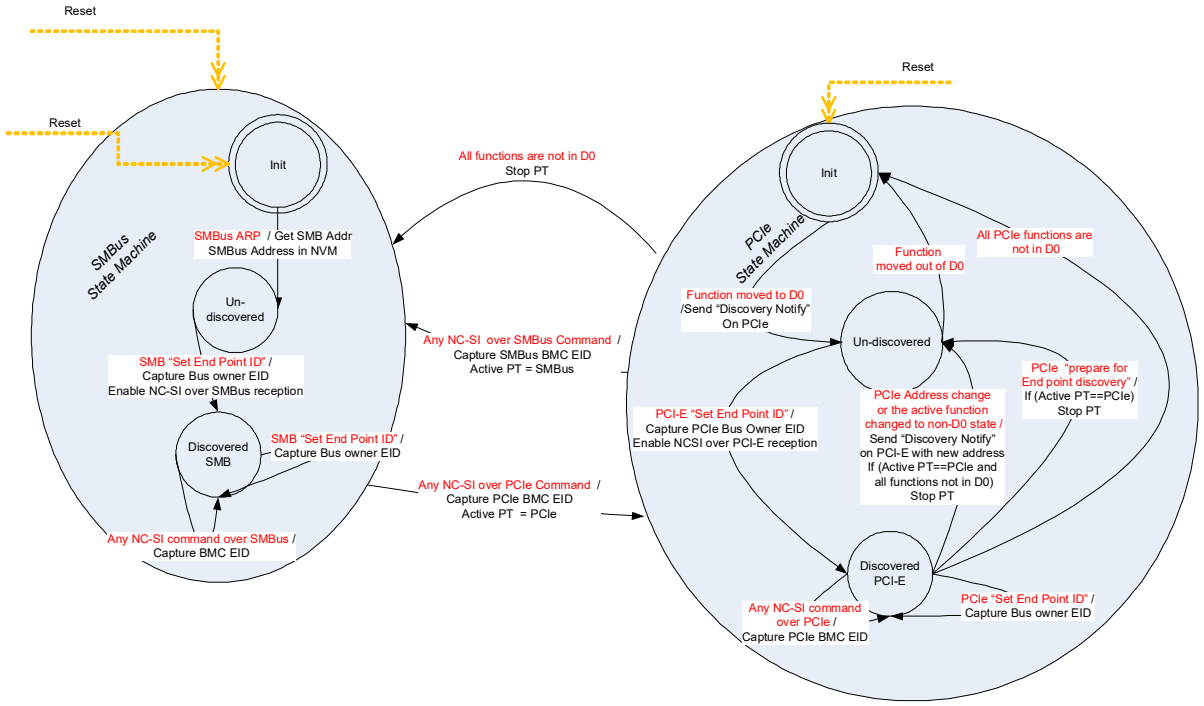


Figure 12-11. MCTP Bus Transition State Machine

12.7.1.3.1 Initial Assignment Flow

1. 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.
2. 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.

3. 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 Slave 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.
4. 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.
5. 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.
6. After the NC-SI channels are enabled, traffic might be sent using the MC and NIC addresses previously discovered.
7. 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.

12.7.1.3.2 SMBus-to-PCIe Transition

1. 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.
2. If the NIC or the MC detects that the PCIe bus is available by detecting a non-zero bus number as reflected in the PF_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.
3. 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.
4. 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.
5. 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 pass-through packets from the moment it sent the first NC-SI command on the PCIe and the moment a response is received for this command.
6. The transition of NC-SI traffic (pass-through or commands/responses) from SMBus to PCIe should be done on a packet boundary and should not interrupt a packet fragmentation or reassembly.

12.7.1.3.3 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:

1. The endpoint should send a Discovery Notify MCTP command on the PCIe bus using the new Requester ID.
2. 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.
3. The bus owner sends an Routing Information Update command to all supporting endpoints that might then update the parameters of their counterpart they use.

12.7.1.3.4 PCIe-to-SMBus Transition

1. 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 pass-through traffic or NC-SI traffic.
2. 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 pass-through traffic or NC-SI traffic.
3. Upon detection of the unavailability of the PCIe bus, the MC transitions the NC-SI channel to the MCTP over SMBus as previously described.

Note: The transition of NC-SI traffic (pass-through 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.

12.7.2 MCTP over PCIe

12.7.2.1 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 12-38. 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	Attr[2]	R	TH[2]	TD[2]	EP[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								SOM	EOM	SEQ#			TO	Tag	
IC	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.

12.7.2.2 PCIe Discovery Process

The E810 follows the discovery process described in Section 5.9 of the MCTP PCIe VDM Transport Binding Specification (DSP0238).

After receiving an endpoint discovery message (while in undiscovered stage), the E810 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 E810 changes due to a re-enumeration of the bus, the E810 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.

12.7.2.3 MCTP over PCIe Special Features

The E810 supports the following optional features of MCTP when running over PCIe:

- Rate limiting
- ACLs

12.7.2.3.1 MCTP Uplink Rate Limiting

As the PCIe link can carry a traffic bandwidth much higher than what the MC can sustain, to avoid drop of packets, the E810 allows rate limiting of the MCTP pass-through traffic. The E810 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.
- **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.

12.7.2.3.2 Service Provider MCTP Endpoint ACLs

The E810 supports a set of ACLs that allows reception of sensitive commands only from a specific bus number (in the Requester ID). The device and function part of the Requester ID are ignored for this purpose.

If ACLs are enabled (by clearing the *Disable ACLs* NVM bit) the following flow is used decide which packets are accepted.

Commands can be divided to three types:

- ACL programming commands: Such commands can be received only from the address that sent the Prepare For Endpoint Discovery command via broadcast routing.
- Sensitive commands including all the NC-SI commands and PT traffic. These commands can be received only from requesters whose bus number is set in the ACL list. If an MCTP packet is dropped, the SPMEACLD counter is increased. This counter can be read by the MCTP bus owner using the Get ACL Violation Counters command.
- Regular MCTP commands are received from any requester. However, the Set EID command is processed only if received from the address that sent the Prepare For Endpoint Discovery command via broadcast routing.

The E810 supports four ACL entries.

Note: This feature is disabled by default in NVM, as it requires a system infrastructure supporting the feature.

12.7.3 MCTP over SMBus

The message format used for NC-SI over MCTP over SMBus is as follows:

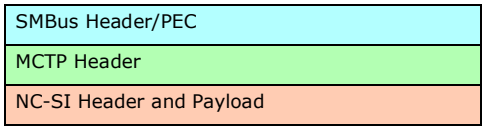


Table 12-39. NC-SI/Ethernet over MCTP over SMBus Message Format

+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 Slave Address							0	Command Code = MCTP = 0Fh								Byte Count								Source Slave Address							1
MCTP Reserved				Header Version = 1				Destination Endpoint ID								Source Endpoint ID								SOM		EOM		SEQ#		TO	Tag
IC[1]				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.

12.7.3.1 SMBus Discovery Process

The E810 follows the discovery process described in Section 6.5 of the MCTP SMBus/I^2C Transport Binding Specification (DSP0237). It indicates support for ASF in the SMBus getUID command (see Section 12.3.2.1.3.2). It responds to any SMBus command using the MCTP command code. This ensures that the bus owner knows the E810 supports MCTP.

Note: MCTP commands over SMBus are received from any master address and are answered to the sender. There is no capturing of the bus owner address from any specific command.

12.7.3.2 MCTP over SMBus Special Features

The E810 supports the following optional features of MCTP when running over SMBus:

- Fairness arbitration

12.7.3.2.1 Fairness Arbitration

When sending MCTP messages over SMBus and when fairness arbitration is enabled (see Section 6.3.58.3), the E810 should adhere to the fairness arbitration as defined in Section 5.13 of DSP0237 when sending MCTP messages.

12.7.4 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 E810 supports both messages. When used only for control, only the NC-SI (0x2) message type is supported.

In addition to the previous message types supported by the E810, the PCIe-based VDM message type is also supported over PCIe to support ACL commands.

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 E810 support for NC-SI over MCTP is similar to the support for NC-SI over RBT with the following exceptions:

- A set of new NC-SI OEM commands used to expose the NC-SI over MCTP capabilities.
- The format of the packets is modified to account for the new transport layer as described in the sections that follow.

12.7.4.1 NC-SI to MCTP Mapping

The eight network ports of the E810 (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 Master Enable* bit of at least one of the functions is set, the endpoint is mapped to the first available function.
2. If the *Bus Master 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 slave address used for the SMBus endpoint is the slave address of the first port.

Section 12.7.1.2 describes the transition between the two buses.

Both endpoints (SMBus and PCIe) might be active concurrently. However, pass-through traffic can be transferred only through one of them. If the PCIe endpoint is active, it is used for pass-through 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 E810 should expect MCTP commands from two sources: the bus owner and the MC. In addition, it should expect pass-through 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 12-12.

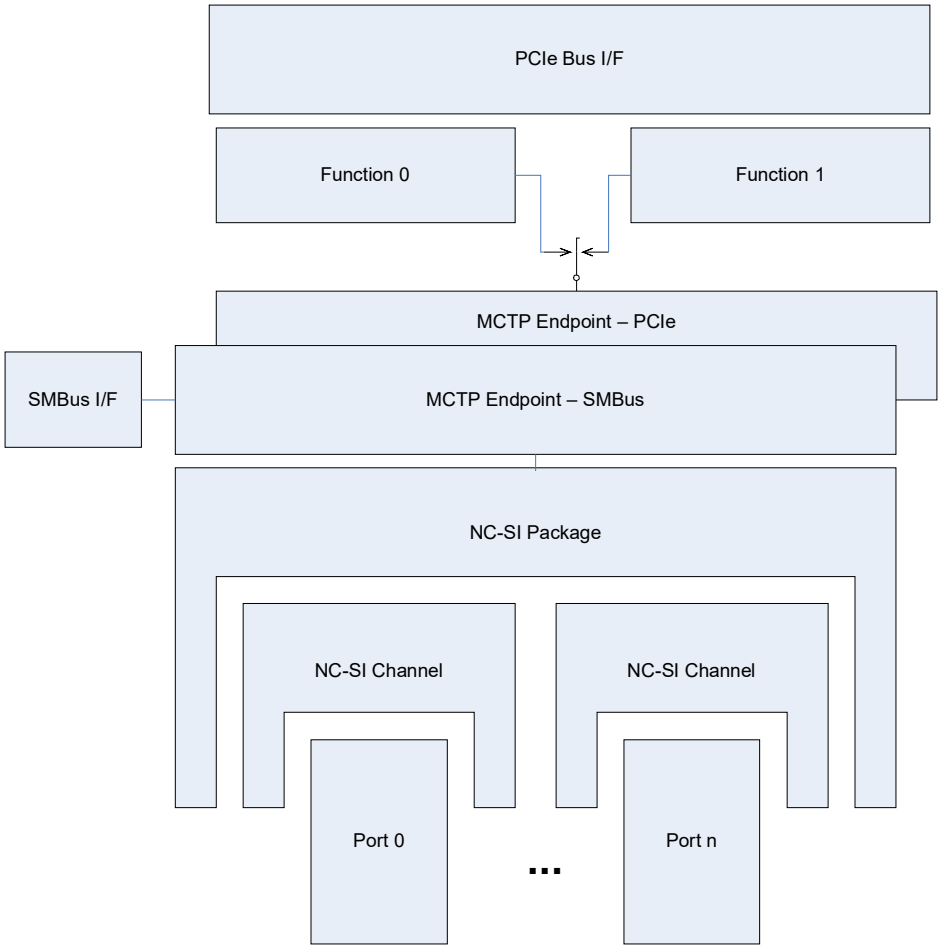


Figure 12-12. MCTP Endpoints Topology

12.7.4.2 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.

12.7.4.2.1 Control Packets

The format used for control packets (commands, responses, and AENs) is as follows:

Table 12-40. 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																Checksum															
Checksum																															

1. The channel ID is defined as described in Section 12.3.2.2

Note: The MAC Header and MAC FCS present when working over NC-SI are not part of the packet in MCTP mode.

12.7.4.2.2 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 E810. In transmit, the CRC is added by the E810.

Table 12-41. 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								EtherType																Ethernet Packet							
Ethernet packet																															
....																															
....																															

12.7.5 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.

12.7.6 OEM Commands

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.

12.7.7 MCTP Programming

The MCTP programming model is based on:

- 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 12.7.7.1.

- A subset of the NC-SI commands used in the regular NC-SI interface, including all the OEM commands as described in Section 12.6.2 (NC-SI programming I/F). The specific commands supported are listed in Table 12-13 and Table 12-16.

Note: For all MCTP commands (both native MCTP commands and NC-SI over MCTP), the response uses the Msg tag received in the request with *TO* bit cleared.

12.7.7.1 MCTP Commands Support

Table 12-42 lists the MCTP commands supported by the E810.

Table 12-42. MCTP Commands Support

Command Name	Command Code	General Description	E810 Support as Initiator	E810 Support as Responder	Section Reference
Reserved	0x00	Reserved.	---	---	---
Set Endpoint ID	0x01	Assigns an EID to the endpoint at the given physical address.	N/A	Yes	12.7.7.1.1
Get Endpoint ID	0x02	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 12.7.7.1.2 for details.	No	Yes	12.7.7.1.2
Get Endpoint UUID	0x03	Retrieves a per-device unique UUID associated with the endpoint. See Section 12.7.7.1.3 for details.	No	Yes	12.7.7.1.3
Get MCTP Version Support	0x04	Lists which versions of the MCTP control protocol are supported on an endpoint. See Section 12.7.7.1.4 for details.	No	Yes	12.7.7.1.4
Get Message Type Support	0x05	Lists the message types that an endpoint supports. See Section 12.7.7.1.5 for details.	No	Yes	12.7.7.1.5
Get Vendor Defined Message Support	0x06	Used to discover an MCTP endpoint's vendor specific MCTP extensions and capabilities. See Section 12.7.7.1.6 for details.	No	Yes	12.7.7.1.6
Resolve Endpoint ID	0x07	Used to get the physical address associated with a given EID.	Yes	N/A	---
Allocate Endpoint IDs	0x08	Used by the bus owner to allocate a pool of EIDs to an MCTP bridge.	N/A	N/A	---
Routing Information Update	0x09	Used by the bus owner to extend or update the routing information that is maintained by an MCTP bridge.	N/A	N/A	---
Get Routing Table Entries	0x0A	Used to request an MCTP bridge to return data corresponding to its present routing table entries.	No	N/A	---
Prepare for Endpoint Discovery	0x0B	Used to direct endpoints to clear their discovered flags to enable them to respond to the Endpoint Discovery command.	N/A	Yes[1]	---
Endpoint Discovery	0x0C	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]	---
Discovery Notify	0x0D	Used to notify the bus owner that an MCTP device has become available on the bus.	Yes[1]	N/A	---
Get Network ID	0x0E	Used to get the MCTP network ID	No	No	---
Query Hop	0x0F	Used to discover what bridges, if any, are in the path to a given target endpoint and what transmission unit sizes the bridges pass for a given message type when routing to the target endpoint.	No	No	---

Table 12-42. MCTP Commands Support [continued]

Command Name	Command Code	General Description	E810 Support as Initiator	E810 Support as Responder	Section Reference
Query Rate Limit	0x11	Used to discover the data rate limit settings of the given target for incoming messages.	No	Yes	12.7.7.1.7
Request Tx Rate Limit	0x12	Used to request the allowed transmit data rate limit for the given endpoint for outgoing messages.	No	Yes	12.7.7.1.8
Update Rate Limit	0x13	Used to update the receiving side on change to the transmit data rate that was not requested by the receiver.	No	Yes	12.7.7.1.9
Query Supported Interfaces	0x14	Used to discover the existing device MCTP interfaces.	No	Yes	12.7.7.1.10

1. These commands are supported only for MCTP over PCIe.

12.7.7.1.1 Set Endpoint ID (0x01)

The E810 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 E810 can be set only through their own interface, Set EID and Force EID are equivalent. The Reset EID operation is not supported by the E810.

The Set Endpoint ID response of the E810 is described in Table 12-43.

Table 12-43. Set Endpoint ID Response

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.

12.7.7.1.2 Get Endpoint ID (0x02)

The Get Endpoint ID response of the E810 is listed in Table 12-44.

Table 12-44. Get Endpoint ID Response

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	0x00 = PCIe 0x01 = SMBus — Fairness arbitration protocol supported.

12.7.7.1.3 Get Endpoint UUID (0x03)

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.

12.7.7.1.4 Get MCTP Version Support (0x04)

Table 12-45 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 in Section 12.7.7.1.5.

Table 12-45. Get MCTP Version Support Returned Value

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 Unsupported 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		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.

12.7.7.1.5 Get Message Type Support (0x05)

The Get Message Type Support response of the E810 is listed in Table 12-46.

Table 12-46. Get Message Type Support Response

Bytes	Description	Value
1	Completion Code	0x00
2	MCTP Message Type Count	0x01/0x02/0x03 — The E810 supports up to four additional message types, depending on the mode of operation and the bus used.
3:5	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.
		0x7E (PCIe based VDM messages) — Over PCIe. Over SMBus also if OEM commands are supported.

12.7.7.1.6 Get Vendor-Defined Message Support (0x06)

The Get Vendor Defined Message Support response of the E810 is listed in Table 12-47 if *Vendor ID Set Selector* equals 0x00.

Table 12-47. Get Vendor-Defined Message Support Response

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)

12.7.7.1.7 Query Rate Limit (0x11)

When receiving a Query Rate Limit command over PCIe, the device should respond with the following parameters:

Table 12-48. Query Rate Limit Response

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 1 Gb/s in 64-bytes packets.
10:13	Transmit Rate limiter capabilities — Maximum supported rate limit.	0x1DCD65 — Reflects a rate of 1 Gb/s in 64-bytes packets.
14:17	Transmit Rate limiter capabilities — Minimum supported rate limit.	0x7A1 — Reflects a rate of 1 Mb/s 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.

Table 12-48. Query Rate Limit Response [continued]

Offset	Description	Value
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.

12.7.7.1.8 Request Tx Rate Limit (0x12)

12.7.7.1.8.1 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 is returned.

The setting of the rate limit is done as described in Section 3.5.4.5 "MCTP over PCIe Rate Limiter Configuration" of hardware/firmware interface document.

12.7.7.1.8.2 Request Tx Rate Limit Send

The device currently does not require a rate limit from the MC, assuming the supported bandwidth of 1 Gb/s is large enough to absorb any MC traffic.

12.7.7.1.9 Update Rate Limit (0x13)

The device never sends this command and completes it without error, but ignores it when received.

12.7.7.1.10 Query Supported Interfaces (0x14)

The 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.

Table 12-49. Query Supported Interfaces Response

Offset	Description	Value
2	Supported Interfaces Count	1 or 2 according to the available interfaces.
3	First interface Type	0x01 if SMBus speed is 100 KHz. 0x04 if SMBus speed is 400 KHz. 0x05 if SMBus speed is 1 MHz.
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.

Note: Bytes 3:4 are returned if SMBus is available. Bytes 5:6 are returned if PCIe is available.

12.8 PLDM Support

The E810 supports PLDM over MCTP protocol. Within PLDM, the following data types are supported:

- 0x0 - PLDM Messaging Control and Discovery
- 0x2 - PLDM Monitoring and Control
- 0x5 - PLDM Firmware Update

This section does not include details of the PLDM specification itself. Rather, 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:

- DSP2040: 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
- oDSP0218: Platform Level Data Model (PLDM) for Redfish Device Enablement

12.8.1 PLDM Base Implementation

PLDM is supported over PCIe or over SMBus. The packet format is described in Table 12-50:

Table 12-50. 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 and MCTP header field is as defined in the specification. The values of the PLDM fields are described in Table 12-51.

Table 12-51. 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.

12.8.1.1 Reset Conditions

The PLDM state is reset when the MCTP layer below it is reset. In the case of PCIe VDM, this is EMPR and PERST reset events.

12.8.1.2 PLDM Control Commands

The following messages (as defined in the DSP0240 PLDM Base Specification) should be supported. For these messages, the PLDM type is zero.

Table 12-52. 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.	12.8.1.2.1
GetPLDMTypes	0x04	Return supported PLDM types.	12.8.1.2.2
GetPLDMCommands	0x05	Return supported PLDM commands per type.	12.8.1.2.3

12.8.1.2.1 GetPLDMVersion (0x03)

The GetPLDMVersion command returns the supported versions of PLDM types.

The command and response are described in Table 12-53.

The command is supported for PLDM type 0 (PLDM Messaging Control and Discovery) and type 5 (PLDM Firmware Update).

Table 12-53. 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:
		{
		GetNextPart = 0x00,
		GetFirstPart = 0x01
		}
		Accepted only if *GetFirstPart*.

Byte	**Type**	**Response Data**

0	enum8	**CompletionCode**
		Possible values:
		{
		PLDM_BASE_CODES,
		INVALID_DATA_TRANSFER_HANDLE = 0x80,
		INVALID_TRANSFER_OPERATION_FLAG = 0x81,
		INVALID_PLDM_TYPE_IN_REQUEST_DATA = 0x83
		}

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 0xF1F0F100 (1.0.1)

12.8.1.2.2 GetPLDMType (0x04)

The GetPLDMType command returns the supported PLDM types.

The command and response are described in Table 12-54.

Table 12-54. 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.

12.8.1.2.3 GetPLDMCommand (0x05)

The GetPLDMCommand command returns the supported command of PLDM per type.

The command and response are described in Table 12-55.

Table 12-55. 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, and 5. 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 = 5: 0xF1F0F000 (1.0.0) Otherwise returns an INVALID_PLDM_VERSION_IN_REQUEST_DATA.
Byte	**Type**	**Response Data**

0	enum8	**CompletionCode**
		Possible values:
		{
		PLDM_BASE_CODES,
		INVALID_PLDM_TYPE_IN_REQUEST_DATA = 0x83,
		INVALID_PLDM_VERSION_IN_REQUEST_DATA = 0x84
		}

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 12.8.4 (SetTID & GetTID are not included). For type 5, return 0x0000_0000_0000_0000_3C19_0006 (according to the supported commands in Section 12.8.4. For type 6, return 0x0003_0000_005F_003E (according to the supported commands in Section 12.8.4). ***Note:*** The commands reported must be according to the firmware's actual support for the commands where NC is the responder.

12.8.2 PLDM Monitoring and Control Support

12.8.2.1 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 12-56. PLDM Monitoring and Control Commands

Command Code	Command	Supported?	Section Reference
0x01	SetTID (see DSP0240)	Yes	---
0x02	GetTID (see DSP0240)	Yes	---
0x03	GetTerminusUID	Yes	12.8.2.1.1
0x04	SetEventReceiver	Yes	12.8.2.1.2
0x05	GetEventReceiver	Yes	12.8.2.1.3
0x0A	PlatformEventMessage	Yes	12.8.2.1.4
0x0B	PollForPlatformEventMessage	Yes	12.8.2.1.5
0x0C	EventMessageSupported	Yes	12.8.2.1.6
0x0D	EventMessageBufferSize	Yes	12.8.2.1.7
0x10	SetNumericSensorEnable	Yes	12.8.2.1.8
0x11	GetSensorReading	Yes	12.8.2.1.9
0x12	GetSensorThresholds	Yes	12.8.2.1.10
0x13	SetSensorThresholds	Yes	12.8.2.1.11
0x14	RestoreSensorThresholds	No	---
0x15	GetSensorHysteresis	Yes	12.8.2.1.12
0x16	SetSensorHysteresis	No	---
0x17	InitNumericSensor	No	---
0x20	SetStateSensorEnables	Yes	12.8.2.1.13
0x21	GetStateSensorReadings	Yes	12.8.2.1.14
0x22	InitStateSensor	No	---
0x30	SetNumericEffecterEnable	No	---
0x31	SetNumericEffecterValue	No	---
0x32	GetNumericEffecterValue	No	---
0x38	SetStateEffecterEnables	No	---
0x39	SetStateEffecterStates	No	---
0x3A	GetStateEffecterStates	No	---
0x40	GetPLDMEventLogInfo	No	---
0x41	EnablePLDMEventLogging	No	---
0x42	ClearPLDMEventLog	No	---
0x43	GetPLDMEventLogTimestamp	No	---
0x44	SetPLDMEventLogTimestamp	No	---

Table 12-56. 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	12.8.2.1.15
0x51	GetPDR	Yes	12.8.2.1.16
0x52	FindPDR	No	---
0x53	GetPDRRepositorySignature	Yes	12.8.2.1.17
0x58	RunInitAgent	No	---

12.8.2.1.1 GetTerminusUID (0x3)

The GetTerminusUID command returns a UUID of the device. The command and response are described in Table 12-57.

Table 12-57. 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 12-58.

Table 12-58. 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	
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.

12.8.2.1.2 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 12-59.

Table 12-59. 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 *eventMessageGlobalEnable* = enable, is 0x00 (MCTP), ignored otherwise. The command is rejected and an INVALID_PROTOCOL_TYPE *completionCode* returned if the *transportProtocolType* is incorrect.
uint8	**eventReceiverAddressInfo** Reflects the EID of the event receiver, and should be used to create events messages. Valid only if *eventMessageGlobalEnable* 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.

12.8.2.1.3 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 12-60.

Table 12-60. 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.

12.8.2.1.4 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 12-61:

Table 12-61. 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 *eventClass*.
Type	**Response Data**

enum8	**completionCode**
	Possible values:
	{
	PLDM_BASE_CODES,
	UNSUPPORTED_EVENT_FORMAT_VERSION = 0x81
	}

enum8	Status - can be ignored.

12.8.2.1.5 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 12-62. 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 *TransferOperationFlag* is set to GetFirstPart. If *TransferOperationFlag* is set to GetNextPart, *dataTransferHandle* should be equal to the *nextDataTransferHandle* 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.
Type	**Response Data**

enum8	**completionCode**
	Possible values:
	{
	PLDM_BASE_CODES,
	UNSUPPORTED_EVENT_FORMAT_VERSION = 0x81,
	EVENT_ID_NOT_VALID = 0x82
	}

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 is 0x0000 is returned if the terminus internal event queue is empty. If *TransferOperationFlag* in the request message was set to AcknowledgementOnly and the event queue is non-empty, the terminus supplies sentinel value 0xFFFF for this field. *eventID* value generated by firmware is increased for each new event (starting from 1).
All of the following fields are returned only if *eventID* 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 this response represents. Possible values: 0x1 = Start 0x2 = Middle 0x4 = End 0x5 = StartAndEnd

Table 12-62. PollForPlatformEventMessage Command and Response [continued]

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.
uint32	**eventDataSize** Size in byes of *eventData*.
var	**eventData** Event data based on the *eventClass*.
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.

12.8.2.1.6 EventMessageSupported (0xC)

Table 12-63. 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. Mus be 0b. The returned value is 0x6 (Synchronous and Asynchronous support without heartbeat).

Table 12-63. EventMessageSupported Command and Response [continued]

uint8	**numberEventClassReturned** The count of *eventClass* enumerated bytes returned in this response. Returns 5 - supported events listed below.
uint8	**eventClass [0]** 0x00 = sensorEvent
uint8	**eventClass [1]** 0x02 = redfishTaskExcuteEvent
uint8	**eventClass [2]** 0x03 = redfishMessageEvent
uint8	**eventClass [3]** 0x04 = pldmPDRRepositoryChgEven
uint8	**eventClass [4]** 0x05 = pldmMessagePollEvent

12.8.2.1.7 EventMessageBufferSize (0xD)

Table 12-64. 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.

12.8.2.1.8 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 12-65.

The default at init time should be to have all sensors enabled.

Table 12-65. 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.

Type	**Response Data**

enum8	**completionCode**
	Value:
	{
	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
	}

12.8.2.1.9 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 12-66.

Table 12-66. 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:
	{
	PLDM_BASE_CODES,
	INVALID_SENSOR_ID = 0x80,
	EARM_UNAVAILABLE_IN_PRESENT_STATE = 0x81
	}

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 12-66. GetSensorReading Command and Response [continued]

enum8	**sensorOperationalState**
	The state of the sensor itself
	Possible values:
	{
	enabled,
	disabled,
	unavailable,
	statusUnknown,
	failed,
	initializing,
	shuttingDown,
	inTest
	}
	Relevant values are:
	enabled — Enabled and operating. The sensor is able to return valid *presentState*, *previousState*, *presentReading*, and *eventState* values. This state can be set through the SetNumericSensorEnable command.
	disabled — The sensor is disabled from returning *presentReading* 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 *presentState* or *eventState* values.

enum8	**sensorEventMessageEnable**
	Possible values:
	{
	noEventGeneration,
	eventsDisabled,
	eventsEnabled,
	opEventsOnlyEnabled,
	stateEventsOnlyEnabled
	}
	as set in the SetNumericSensorEnable command

enum8	**presentState** The most recently assessed state value monitored by the sensor. If the sensorOperationalState is set to enabled the sensor must return a value other than "Unknown" for the *presentState*. If the *sensorOperationalState* is not set to enabled, the sensor returns "Unknown" for the *presentState*. See Section 12.8.3.2 for the reported state per sensor.
enum8	**previousState** The state that the *presentState* was entered from. See Section 12.8.3.2 for the reported state per sensor. If the *sensorOperationalState* is not set to enabled, the sensor returns "Unknown" for the *previousState*.
enum8	**eventState** Same as *presentState* 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 12.8.3.2 for the reported value per sensor.

12.8.2.1.10 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 12-67.

Table 12-67. 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,
	NVALID_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 *WarningHigh* field in NVM if Thermal Sensor Thresholds.*upperWarning* 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 *CriticalHigh* field in NVM if Thermal Sensor Thresholds.*upperCritical* 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 *FatalHigh* field in NVM if Thermal Sensor Thresholds.*upperFatal* supported is set, zero otherwise.
uint8	**lowerThresholdWarning** Value: 0
uint8	**lowerThresholdCritical** Value: 0
uint8	**lowerThresholdFatal** Value: 0

12.8.2.1.11 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 12-68.

Table 12-68. 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	
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
	}

12.8.2.1.12 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 12-69.

Table 12-69. 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 *Hysteresis* field in Thermal Sensor Tolerance & Hysteresis NVM word for the relevant sensor.

12.8.2.1.13 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 12-70.

Table 12-70. 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 *OpFields* 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 *compositeSensorCount* sensors The accessed sensors are 0-*compositeSensorCount*-1
compositeSensorCount instances of *OpFields*:	

enum8	**sensorOperationalState**
	The desired state of the sensor. Defines if the sensor is enabled, disabled or unavailable.
	Possible values:
	{
	enabled,
	disabled,
	unavailable
	}
	Default is enabled.

enum8	**EventMessageEnable**
	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
	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.
	enableEvents and *enableStateEventsOnly* are treated the same.

Type	**Response Data**

enum8	**completionCode**
	Possible values:
	{
	PLDM_BASE_CODES
	INVALID_SENSOR_ID = 0x80,
	EVENT_GENERATION_NOT_SUPPORTED = 0x82
	}

12.8.2.1.14 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 12-71.

Table 12-71. 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 12.8.3.2 for the state events reported.
enum8	**previousState** The state that the *presentState* was entered from. See Section 12.8.3.2 for the reported state per sensor. If the *sensorOperationalState* is not set to enabled, the sensor returns "Unknown" for the previousState.
enum8	**eventState** same as *presentState*.

12.8.2.1.15 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 12-72.

Table 12-72. 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 *updateInProgress* (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 12-72. GetPDRRepositoryInfo Command and Response [continued]

uint8	**dataTransferHandleTimeout** The minimum interval, in seconds, that a *dataTransferHandle* 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).

12.8.2.1.16 GetPDR (0x51)

The GetPDRRepositoryInfo is used to access the PDRs in the device. The command and response are described in Table 12-73.

The PDR data structures used to fill the *recordData* in this command are described in Section 12.8.3.3.

Table 12-73. GetPDR Command and Response

Type	Request Data
uint32	**recordHandle** See Section 12.8.3.3 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 *transferOperationFlag* is *GetFirstPart*}

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 *transferOperationFlag* field is set to *GetFirstPart*, set this value to 0x0000. If the *transferOperationFlag* field is set to *GetNextPart*, set this to the *recordChangeNumber* 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 12-73. GetPDR Command and Response [continued]

uint32	**nextRecordHandle** The next record handle is based on Table 12-80, where the handles are returned in ascending values. ***Note:*** When RDE is enabled, an additional list of PRDs is added, as described in Section 12.8.3.3.7. 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 *recordData* bytes returned in this response. Special value: Returns 0x0000 if the *requestCount* was 0x0000.
(var)	**recordData** PDR data bytes. This field is absent if *responseCount*=0x0000. The number of PDR data bytes returned in this field must match *responseCount*.
If *transferFlag* = 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 x8 + x2 + x1 + 1 (This is the same polynomial used in the MCTP over SMBus/I^2C 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 *transferFlag*=End.

12.8.2.1.17 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 12-74. 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

12.8.2.2 PLDM Monitoring and Control Events

The following events can be generated by the E810:

- **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 12-75 and Table 12-76 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 12-75 through Table 12-79 describe the various events formats:

Table 12-75. PDR Repository Change Event Data

Type	Request Data

enum8	**eventDataFormat**
	Possible values:
	{
	refreshEntireRepository,
	formatIsPDRTypes,
	formatIsPDRHandles
	}
	This field indicates if the *changedRecords* 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 *refreshEntireRepository*. To signal that only certain types of PDRs should be refreshed, the device supplies the value *formatIsPDRTypes* and provides one change record below for each type of PDR to be refreshed.

uint8	**numberOfChangeRecords** The number of *changeRecords* (N_R) following this field. If the *eventDataFormat* is *refreshEntireRepository*, this value is zero.
var	**changeRecord [0]** See Table 12-76 for details. This field is not present if the *numberOfChangeRecords* is zero (0).
var	**changeRecord [1]**
...	...
var	**changeRecord [N_R - 1]**

Table 12-76. pldmPDRRepositoryChgEvent changeRecord Format

Type	Request Data

enum8	**eventDataOperation**
	Possible values:
	{
	refreshAllRecords,
	recordsDeleted,
	recordsAdded,
	recordsModified
	}
	For each pldmPDRRepositoryChgEvent 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 *refreshAllRecords* is only supplied when *eventDataFormat* was set to formatIsPDRTypes. 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 enumerated by the *eventDataFormat* field in the pldmPDRRepositoryChgEvent 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 12-77. 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 12-78. redfishMessageEvent Class eventData Format

Type	Request Data
uint8	**eventCount** The number of Redfish Events (N) encoded in the *eventData* field below.
uint16	**eventDataLength** Length in bytes of the *eventData* 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 *eventData* field below.
enum8	**eventSeverity [0]** The severity of the first Redfish Event in the Redfish EventRecords array encoded in *eventData* 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 *eventData* field below.
enum8	**eventSeverity [N - 1]** The severity of the last Redfish Event in the Redfish EventRecords array encoded in *eventData* below. Values: 0 = OK 1 = Warning 2 = Critical
bejEncoding	**eventData** BEJ encoded Event payload data. The *bejEncoding* PLDM type is defined in DSP0218.

Table 12-79. 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.

12.8.2.3 PDR Dynamic Changes Flow

In the OCP 3.0 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.
- Transition from breakout to non-breakout cable configuration and vice versa. See Section 12.8.2.3.1 on the events that indicates these events.

Once a topology change impacting the PDR structure is detected, the following logical flow should be applied:

1. Prepare a separate copy of the new PDR structure.
2. For each record that changed, or that was not present in the previous structure, increase the *recordChangeNumber*.
3. Change the state of the Configuration Change sensor in the NIC composite sensor to Configuration Change Detected.
4. If a GetPDR command is in the middle of reading the PDR structure (the last GetPDR response did not have the *transferFlag* set to End or StartAndEnd), wait until the read is done, or 30 seconds from PDR change.
5. In case of timeout, abort the current transfer.
6. Copy the new PDR to the active one.
7. Update the updateTime used in the GetPDRRepositoryInfo answer to reflect the current time (using the free-running timer of the FW).
8. Update the *RecordCount*, *largestRecordSize*, and *repositorySize* according to the new structure.

12.8.2.3.1 PDR Change Events

12.8.2.3.1.1 QSFP/SFP Plug Events

A module plug event creates a “Media Inserted” event within the firmware. There are two parts of the PDR that can be impacted by this event:

- New plug, cables, and associated sensors additions. These are added immediately.
- New channels. Those are added only if there is no media conflict. Otherwise, they are added only after the next reset after the media conflict is resolved. Added channels create new Link association PDRs.

12.8.2.3.1.2 QSFP/SFP Unplug Events

A module unplug event creates a “Media Removed” event within the firmware. In this case, all the PDRs listed in the previous section are removed.

12.8.2.3.1.3 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

12.8.3 PLDM Monitoring and Control Generic Structures

12.8.3.1 Sensors Numbering

The OCP 3.0 NIC modeling proposal defines a set of IDs for each type of sensor, as in Table 12-80:

Table 12-80. 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
NIC	1	1	100		1100				1
Card Composite State Sensor	1	1			1101	1101	5	5	1
Connectors	4	4	200	203	1105	1108			1
NIC Temp Sensors	10	3			1110	1119	20	29	1
Network Controllers	10	1	1000	1009	1120	1129			1
Network Controller State	1	1			1130	1139	50	59	1
Ports of Network Controller	10	8			1200	1219			1
Link Speed of Network Controller	10	8			1800	1899	600	699	1
Link State of Network Controller	10	8			1900	1999	700	799	1
Temp Sensor per Network Controller	10	3			1300	1399	200	299	1
Plugs	4	4	1010	1013	1250	1253			1
Plug Power Sensor	20	4			1140	1159	250	269	1
Plug Temp Sensor	10	4			1500	1699	300	499	1
Plug Composite Sensor	1	4 (1 per plug)			1400	1419	500	519	1
Plug Entity Association	20	8	1040	1059	1160	1179			1
Channel Entity Association	20	8	1060	1079	1180	1199			21

Calculated
Model Constant
User Parameter

Where:

Item	The Name of the References Items
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.

This maps (after some changes) to the following PDR/Sensor Diagram:

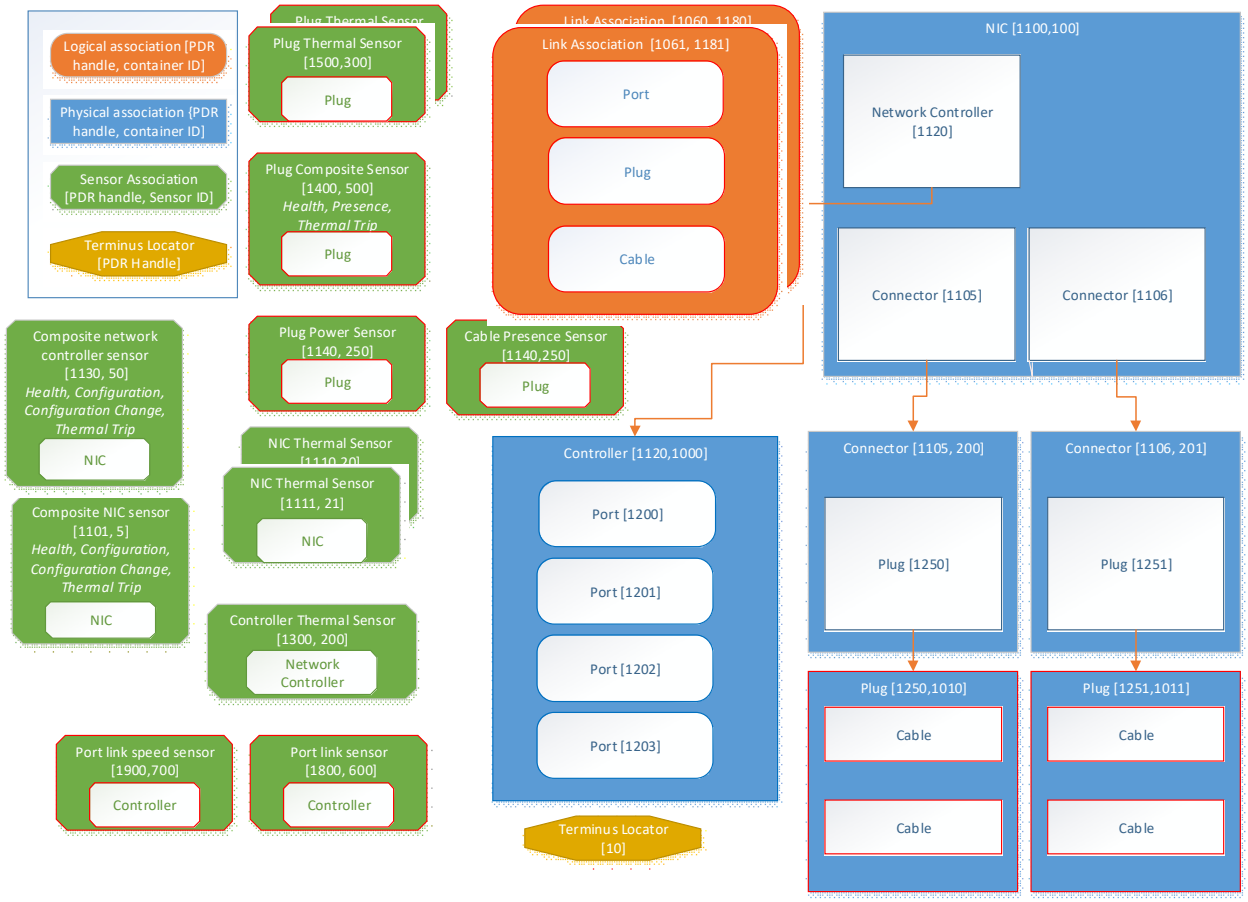


Figure 12-13. PDR/Sensor Diagram

Note: The PDRs with a red border are dynamic and may change due to module plug events or breakout cables insertion events.

12.8.3.2 Sensors

The following sensors are maintained in the device to be reported via the PLDM commands described in Section 12.8.2.1.

- 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 12.8.3.1.

12.8.3.2.1 Numeric Sensors

12.8.3.2.1.1 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 12.8.3.1 (300 +n, 20 +n, 200 for plug, NIC, Controllers sensors respectively).
enum8	**sensorEventClass** = numericSensorState (0x2) or sensorOpState (0x0) If the Sensor has a *sensorEventMessageEnable* value of *enableOpEventsOnly*, return a *sensorOpState*. If the Sensor has a *sensorEventMessageEnable* value of *enableEvents*, return a *numericSensorState*. Otherwise, events are not generated.
enum8	**presentState** The *eventState* value from the state change that triggered the event message. If temperature is >= Fatal Threshold, Return *UpperFatal* (0xA) If temperature is >= Critical Threshold return *UpperCritical* (0x9). If temperature is >= Warning Threshold return *UpperWarning* (0x8) Otherwise, return Normal (0x1). Thresholds are set by the SetSensorThresholds command.
enum8	**previousState** The *eventState* 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 *presentState* if the *previousState* 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 *sensorEventClass* = *numericSensorState* (0x2).
uint8	**presentReading** Temperature. The resolution is according to the resolution defined in the Thermal Sensor PDR (Section 12.8.3.3.6.1) - returned only if *sensorEventClass* = *numericSensorState* (0x2).

12.8.3.2.1.2 Link Speed Data Structure

Type	Request Data
uint16	**sensorID** = 599 + Port#
enum8	**sensorEventClass** = numericSensorState (0x2) or sensorOpState (0x0) If the Sensor has a *sensorEventMessageEnable* value of *enableOpEventsOnly*, return a *sensorOpState*. If the Sensor has a *sensorEventMessageEnable* value of *enableEvents*, return a *numericSensorState*. Otherwise, events are not generated.
enum8	**presentState** Return Normal (0x1).
enum8	**previousState** Return Normal (0x1).
enum8	**sensorDataSize** = uint32 (0x4) - returned only if *sensorEventClass* = *numericSensorState* (0x2).
uint8	**presentReading** Link Speed in Megabits per second - returned only if *sensorEventClass* = *numericSensorState* (0x2).

12.8.3.2.1.3 Plug Power Data Structure

Type	Request Data
uint16	**sensorID** = 250 + Plug#
enum8	**sensorEventClass** = numericSensorState (0x2) or sensorOpState (0x0) If the Sensor has a *sensorEventMessageEnable* value of *enableOpEventsOnly*, return a *sensorOpState*. If the Sensor has a *sensorEventMessageEnable* value of *enableEvents*, return a *numericSensorState*. Otherwise, events are not generated.
enum8	**presentState** Return Normal (0x1).
enum8	**previousState** Return Normal (0x1).
enum8	**sensorDataSize** = uint32 (0x4) - returned only if *sensorEventClass* = *numericSensorState* (0x2).
uint8	**presentReading** Plug nominal power as read from addresses below in the module EEPROM through I^2C 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 *sensorEventClass* = *numericSensorState* (0x2). The value is returned from the sff_get_pwr (activity ID = 0x0027) DNL script.

12.8.3.2.2 NIC Composite State Sensors

12.8.3.2.2.1 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 *eventState* 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.

12.8.3.2.2.2 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 *eventState* 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 the firmware failed to find all the needed TLVs in the PFA or other associated errors.
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. Special value: This value must be set to the same value as *presentState* if the *previousState* 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).

12.8.3.2.2.3 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 *eventState* 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 *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. Special value: This value must be set to the same value as *presentState* if the *previousState* 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).

12.8.3.2.2.4 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 *eventState* 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).

12.8.3.2.3 Controller Composite State Sensors

12.8.3.2.3.1 Controller Health State Data Structure

Type	Request Data
uint16	**sensorID** = 50
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 *eventState* 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 *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. Special value: This value must be set to the same value as *presentState* if the *previousState* 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 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. Special value: This value must be set to the same value as *presentState* if the *previousState* 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).

12.8.3.2.3.2 Controller Configuration Data Structure

Type	Request Data
uint16	**sensorID** = 50
enum8	**sensorEventClass** = stateSensorState (0x1)
uint8	**sensorOffset** = 0x1 (offset in Controller composite state structure)
enum8	**presentState** The *eventState* value from the state change that triggered the event message. One of the following: 1 = Valid Configuration 2 = Invalid Configuration Invalid configuration means the firmware failed to find all the needed TLVs in the PFA or other associated errors.
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. Special value: This value must be set to the same value as *presentState* if the *previousState* 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).

12.8.3.2.3.3 Controller Configuration Change Data Structure

Type	Request Data
uint16	**sensorID** = 50
enum8	**sensorEventClass** = stateSensorState (0x1)
uint8	**sensorOffset** = 0x2 (offset in Controller composite state structure)
enum8	**presentState** The *eventState* 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 *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. Special value: This value must be set to the same value as *presentState* if the *previousState* 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).

12.8.3.2.3.4 Controller Thermal Trip State Data Structure

Type	Request Data
uint16	**sensorID** = 50
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 *eventState* value from the state change that triggered the event message. Return 0x1 (Normal).

12.8.3.2.4 Plug Composite State Sensors

12.8.3.2.4.1 Plug Health State Data Structure

Type	Request Data
uint16	**sensorID** = 500
enum8	**sensorEventClass** = stateSensorState (0x1)
uint8	**sensorOffset** = 0x0 (offset in Plug composite state structure)
enum8	**presentState** The *eventState* 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 *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. Special value: This value must be set to the same value as *presentState* if the *previousState* 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).

12.8.3.2.4.2 Plug Presence State Data Structure

Type	Request Data
uint16	**sensorID** = 500
enum8	**sensorEventClass** = stateSensorState (0x1)
uint8	**sensorOffset** = 0x1 (offset in Plug composite state structure)
enum8	**presentState** The *eventState* value from the state change that triggered the event message. Return 0x1 (Present) if a the plug is connected. Return 0x2 (Not Present) if the plug is not connected TBD if should be plug or cable presence.
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. Special value: This value must be set to the same value as *presentState* if the *previousState* 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).

12.8.3.2.4.3 Plug Thermal Trip State Data Structure

Type	Request Data
uint16	**sensorID** = 500
enum8	**sensorEventClass** = stateSensorState (0x1)
uint8	**sensorOffset** = 0x2 (offset in Plug composite state structure)
enum8	**presentState** ***Note:*** All thermal sensing events here relate to the plug internal TS. The *eventState* value from the state change that triggered the event message. Return 0x1 (Normal).
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. Special value: This value must be set to the same value as *presentState* if the *previousState* 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).

12.8.3.2.5 Simple Sensors

12.8.3.2.5.1 Port Link Sensor

Type	Request Data
uint16	**sensorID** = 699+ port#
enum8	**sensorEventClass** = stateSensorState (0x1)
uint8	**sensorOffset** = 0x0 (not part of a composite)
enum8	**presentState** The eventState 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.

12.8.3.2.5.2 Cable Present Sensor (currently not needed)

There is a single sensor per plug even in case of plugout cable.

Type	Request Data
uint16	**sensorID** = 250
enum8	**sensorEventClass** = stateSensorState (0x1)
uint8	**sensorOffset** = 0x0 (not part of a composite)
enum8	**presentState** The *eventState* value from the state change that triggered the event message Return 0x1 (Connected) if a cable is connected. Return 0x2 (Not Connected) if a cable 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.

12.8.3.3 PDRs

The PDRs are numbered according to Section 12.8.3.1, "Sensors Numbering". The link list used to create the Next Handle Record, which should be created based on the exposed PDRs.

12.8.3.3.1 Terminus Locator PDR

Type	PDR Fields
uint32	**recordHandle** = 0x0000_0010
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
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)
terminusLocatorValue for *terminusLocatorType* = MCTP_EID:	
uint8	**EID** = The MCTP EID that is assigned to the device for the relevant medium.

12.8.3.3.2 Physical Association PDRs

The PDRs are exposed in a high-level representation for the association PDRs.

12.8.3.3.2.1 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	
Container Entity Container ID	100	
Contained Entity - Connector		
Entity Type	185	Connector
Entity Instance Number	1	
Container Entity Container ID	100	
Contained Entity - Connector		
Entity Type	185	Connector
Entity Instance Number	2	
Container Entity Container ID	100	

12.8.3.3.2.2 Controller Association PDR

The controller includes all its ports. There can be up to eight ports for the E810. The actual number is derived from the link topology. The example below assumes two ports.

Network Controller Association PDR		
Container ID	1000	
Container Entity		
Entity Type	144	Network Controller
Entity Instance Number	1	
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	
Contained Entity - Connector		
Entity Type	6	Communication Port
Entity Instance Number	2	
Container Entity Container ID	1000	

12.8.3.3.2.3 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	199 + n	
Container Entity		
Entity Type	185	Connector
Entity Instance Number	n	
Container Entity Container ID	100	
Association Type	Physical to Physical containment	
Contained Entity - Plug		
Entity Type	189	Plug
Entity Instance Number	1	
Container Entity Container ID	199 + n	

12.8.3.3.2.4 Plug Association PDR

A plug association PDR exists per cable per cage of the device, as reflected from the topology file. In case of 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. In case of 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	
Container Entity		
Entity Type	189	Plug
Entity Instance Number	n	
Container Entity Container ID	199 + n	
Association Type	Physical to Physical containment	
Contained Entity - Cable #1		
Entity Type	187	Cable
Entity Instance Number	1	
Container Entity Container ID	1009 + n	
Contained Entity - Cable #2 - Breakout Only		
Entity Type	187	Cable
Entity Instance Number	2	
Container Entity Container ID	1009 + n	
Contained Entity - Cable #3 - Breakout Only		
Entity Type	187	Cable
Entity Instance Number	3	
Container Entity Container ID	1009 + n	
Contained Entity - Cable #4 - Breakout Only		
Entity Type	187	Cable
Entity Instance Number	4	
Container Entity Container ID	1009 + n	

12.8.3.3.3 Logical Association PDRs

The PDRs are exposed in a high-level representation for the association PDRs.

12.8.3.3.3.1 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. In case of 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	1060	
Container Entity		
Entity Type	6	Communication Channel
Entity Instance Number	1	
Container Entity Container ID	1000	Controller
Association Type	Logical to Physical containment	
Contained Entity - Plug		
Entity Type	189	Plug
Entity Instance Number	n	
Container Entity Container ID	1060	
Contained Entity - Cable		
Entity Type	187	Cable
Entity Instance Number	k	Number of the cable within the plug
Container Entity Container ID	1060	

12.8.3.3.4 Composite State Sensor PDRs

12.8.3.3.4.1 NIC Composite State Sensor

NIC Composite State Sensor		
Entity Type	68	Add-in Card
Entity Instance Number	1	
Container Entity Container ID	100	
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	

12.8.3.3.4.2 Controller Composite State Sensor

Network Controller Composite State Sensor		
Entity Type	68	Add-in Card
Entity Instance Number	1	
Container Entity Container ID	100	
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	

12.8.3.3.4.3 Plug Composite State Sensor

Plug #n Composite State Sensor		
Entity Type	189	Port
Entity Instance Number	1	
Container Entity Container ID	1009 + n	
Terminus Handle	0	
Sensor ID	499 + n	
Composite Sensor Count	1	
Sensor Type	1	Health State
Possible States	1=Normal, 3=Critical, 5=Upper_Non_Critical, 7=Upper Critical	
Sensor Type	13	Presence
Possible States	1=Present, 2=Not_Present	
Sensor Type	21	Thermal Trip
Possible States	1=Normal, 2=Over-Temp Shutdown	

12.8.3.3.5 Simple State Sensors PDRs

12.8.3.3.5.1 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 12.8.3.3.2.2)
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	

12.8.3.3.5.2 Cable Presence State

Port #n Cable Presence State Sensor		
Entity Type	189	Plug
Entity Instance Number	1	
Container Entity Container ID	1009 + n	
Terminus Handle	0	
Sensor ID	1139 + n	
Composite Sensor Count	1	
Sensor Type	13	Presence State (Cable)
Possible States	1=Connected, 2=Disconnected	

12.8.3.3.6 Numeric Sensors PDRs

12.8.3.3.6.1 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 12.8.3.1, "Sensors Numbering" (thermal sensors).
uint8	**PDRHeaderVersion** = 0x01
uint8	**PDRType** = 0x2 (Numeric Sensor PDR)
uint16	**recordChangeNumber** According to Section 12.8.2.3, "PDR Dynamic Changes Flow".
uint16	**dataLength** = 0x43 (67)
uint16	**PLDMTerminusHandle** = 0x0001
uint16	**sensorID** According to Section 12.8.3.1, "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 = 0xBD (Physical | Plug (189))
uint16	**entityInstanceNumber** Sensor Number (starting from 0)
uint16	**containerID** According to the container of the thermal Sensor in Section 12.8.3.1, "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)

Type	PDR Fields
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)
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.
birfield8	**supportedThresholds** Return 0x7 (*upperThresholdFatal*, *upperThresholdCritical*, *upperThresholdWarning*)
birfield8	**thresholdAndHysteresisVolatility** Return 0x3 (PLDM subsystem power up, Initialization Agent controller restart/update). TBD if others also.
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 *maxReadable* in Node Thermal Configuration Section.
uint8	**minReadable** As read from *minReadable* in Node Thermal Configuration Section.
enum8	**rangeFieldFormat** = 0x2 (uint16)
birfield8	**rangeFieldSupport** Return 0x2A (normalMax, criticalHigh, fatalHigh supported).
uint16	**nominalValue** = Return 0x0
uint16	**normalMax** As read from *normalMax* 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 *warningHigh* field in Node Thermal Configuration Section.
uint16	**warningLow** = Return 0x0

Type	PDR Fields
uint16	**criticalHigh** As read from *criticalHigh* field in Node Thermal Configuration Section. Value is in degrees C, should be adapted according to the resolution field.
uint16	**criticalLow** = Return 0x0
uint16	**fatalHigh** As read from *fatalHigh* field in Node Thermal Configuration Section.
uint16	**fatalLow** = Return 0x0

12.8.3.3.6.2 Plug Power Sensors

Type	PDR Fields
uint32	**recordHandle** = 1139 + plug number (1..n)
uint8	**PDRHeaderVersion** = 0x01
uint8	**PDRType** = 0x2 (Numeric Sensor PDR)
uint16	**recordChangeNumber** According to Section 12.8.2.3, "PDR Dynamic Changes Flow".
uint16	**dataLength** = 0x47 (71)
uint16	**PLDMTerminusHandle** = 0x0001
uint16	**sensorID** = 249 + plug number (1..n)
uint16	**entityType** = 0xBD (Physical | Plug (189))
uint16	**entityInstanceNumber** = 1
uint16	**containerID** = 1009 + port# (1..n)
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 a the max 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).
real32	**Resolution** = 0x0
real32	**Offset** = 0x0

Type	PDR Fields
uint16	**Accuracy** = 0x0
uint8	**plusTolerance** = 0x0
uint8	**minusTolerance** = 0x0
uint32	**Hysteresis** = 0x0 (sensor does not use hysteresis)
birfield8	**supportedThresholds** = 0x0 (no threshold is supported)
birfield8	**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 *enabledState* 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)
birfield8	**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

12.8.3.3.6.3 Link Speed Sensors

Type	PDR Fields
uint32	**recordHandle** = 1899 + port number (1..n)
uint8	**PDRHeaderVersion** = 0x01
uint8	**PDRType** = 0x2 (Numeric Sensor PDR)
uint16	**recordChangeNumber** According to Section 12.8.2.3, "PDR Dynamic Changes Flow".
uint16	**dataLength** = 0x47 (71)
uint16	**PLDMTerminusHandle** = 0x0001
uint16	**sensorID** = 599 + port number (1..n)
uint16	**entityType** = 0x06 (Physical | Port (6))
uint16	**entityInstanceNumber** Port number (the ordering number of the port reported in controller association PDR - Section 12.8.3.3.2.2, "Controller Association PDR" this link refers to).
uint16	**containerID** = 1000

Type	PDR Fields
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)
birfield8	**supportedThresholds** = 0x0 (no threshold is supported)
birfield8	**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 *enabledState* 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)
birfield8	**rangeFieldSupport** = 0x1 (nominalValue 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

12.8.3.3.7 Redfish PDRs Summary

Table 12-81 defines the PDRs that corresponding to the supported Schemas.

Table 12-81. PDRs corresponding to Supported Schema

Name	Type	Handle	ResourceID	Container	Profile	Section Reference
NetworkAdapter	Resource	4001	1	NetworkAdapters	ACD	12.8.3.3.8.1
NetworkInterface	Resource	4005	5	NetworkInterfaces	ACD	12.8.3.3.8.2
NetworkPorts	Resource	4010	10	1	ACD	12.8.3.3.8.3
NetworkDeviceFunctions	Resource	4020	20	1	ACD	12.8.3.3.8.4
NetworkPort	Resource	4100	100-107 110-117	10	ACD	12.8.3.3.8.5
NetworkDeviceFunction	Resource	4200	200-207 210-217	20	ACD	12.8.3.3.8.6
PCIeFunctions	Resource	4030	30	1	ACD	12.8.3.3.8.7
PCIeFunction	Resource	4300	300-307	3	ACD	12.8.3.3.8.8
EthernetInterfaceCollection	Resource	4040	40	System	Other	12.8.3.3.9.1
EthernetInterface	Resource	4400	400-407 410-417	40	Other	12.8.3.3.9.2
ResetSettingsToDefault	Action	3001	2001	-	Other	12.8.3.3.10.1

12.8.3.3.8 ACD Profile PDRs and Links

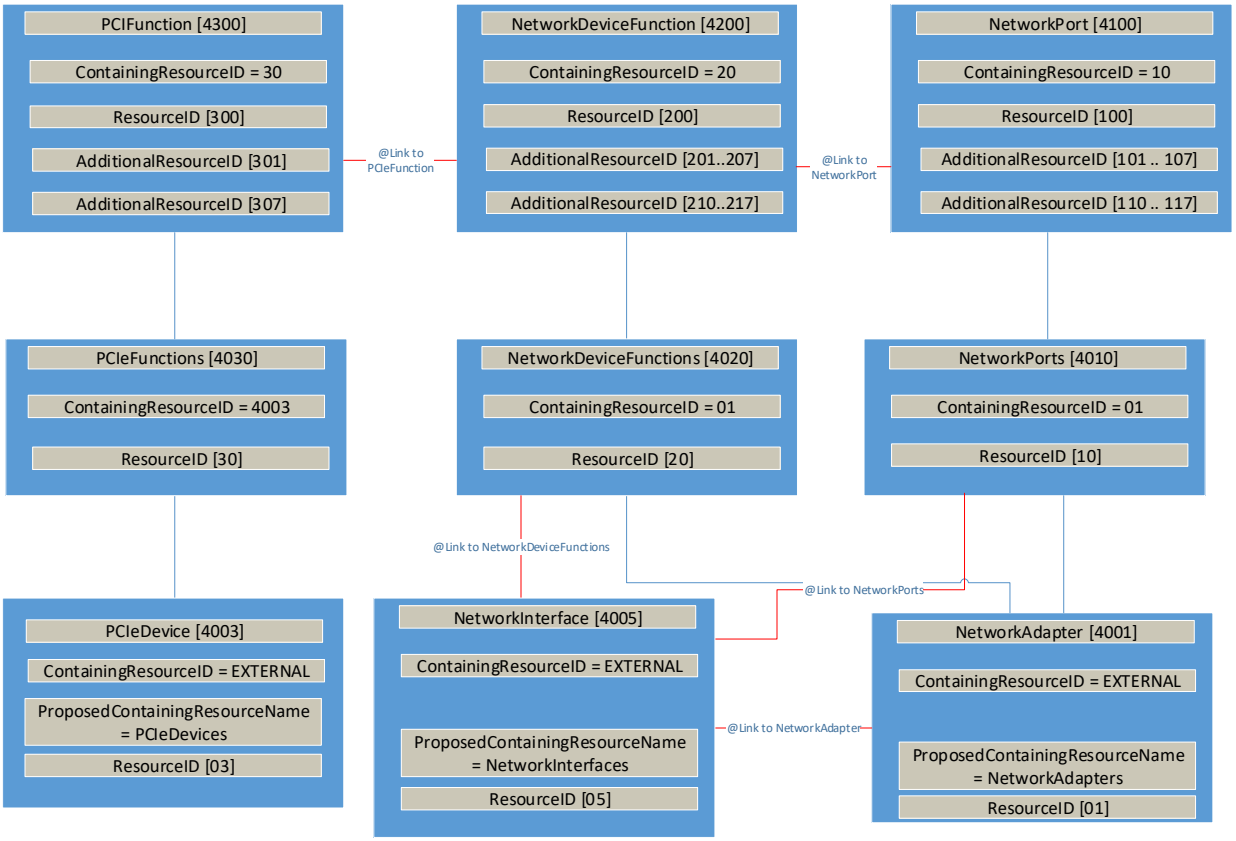


Figure 12-14. ACD Profile PDRs and Links

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.

12.8.3.3.8.1 NetworkAdapter

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	SubURILengthBytes	
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	31
strUTF-8	MajorSchemaName	"NetworkAdapter. NetworkAdapter\0"
uint16	OEMCount	0

12.8.3.3.8.2 NetworkInterface

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.NetworkInterfac eCollection\0"
uint16	SubURILengthBytes	Length in bytes of SubURI
strUTF-8	SubURI	Null
uint16	AdditionalResourceIDCount	0
ver32	MajorSchemaVersion	0xF1F1F100
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

12.8.3.3.8.3 NetworkPorts

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	SubURILengthBytes	Length in bytes of SubURI
strUTF-8	SubURI	"NetworkPorts"
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	44
strUTF-8	MajorSchemaName	"NetworkPortCollection.NetworkPortCollection\0"
uint16	OEMCount	0

12.8.3.3.8.4 NetworkDeviceFunctions

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	SubURILengthBytes	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

12.8.3.3.8.5 NetworkPort

Type	Description	Value
uint32	recordHandle	4100
uint32	ResourceID	100 + Ports[0]
bitfield8	ResourceFlags	2 (is_contained_in_collection)
strUTF-8	ProposedContainingResourceName	"\0"
uint16	SubURILengthBytes	Length in bytes of SubURI
strUTF-8	SubURI	"0"
uint16	AdditionalResourceIDCount	2*PortCount - 1
uint32	AdditionalResourceID [0]	100 + Ports[1]
uint16	AdditionalResourceSubURILengthBytes[0]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[0]	"1"
uint32	...	...
uint32	AdditionalResourceID [PortCount - 2]	100 + Ports[PortCount - 1]
uint16	AdditionalResourceSubURILengthBytes[PortCount - 2]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[PortCount - 2]	"n"
uint32	AdditionalResourceID [PortCount - 1]	110 + PFs[0]
uint16	AdditionalResourceSubURILengthBytes[PortCount - 1]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[PortCount - 1]	"0/Settings"
uint32	...	...
uint32	AdditionalResourceID [2*PortCount - 2]	110 + PFs[PFCount-1]
uint16	AdditionalResourceSubURILengthBytes[PortCount - 2]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[PortCount - 2]	""n"/Settings"
ver32	MajorSchemaVersion	0xF1F2F000
uint16	MajorSchemaDictionaryLengthBytes	Actual length of dictionary stored in NVM.
uint32	MajorSchemaDictionarySignature	32-bit CRC of the schema dictionary.
unit8	MajorSchemaNameLength	24
strUTF-8	MajorSchemaName	"NetworkPort.NetworkPort\0"
uint16	OEMCount	0

12.8.3.3.8.6 NetworkDeviceFunction

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	SubURILengthBytes	Length in bytes of SubURI
strUTF-8	SubURI	"0"
uint16	AdditionalResourceIDCount	2*PFCount-1
uint32	AdditionalResourceID [0]	200 + Ports[1]
uint16	AdditionalResourceSubURILengthBytes[0]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[0]	"1"
uint32	...	...
uint32	AdditionalResourceID [PortCount - 2]	200 + Ports[PortCount - 1]
uint16	AdditionalResourceSubURILengthBytes[PortCount - 2]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[PortCount - 2]	"n"
uint32	AdditionalResourceID [PortCount - 1]	210 + PFs[0]
uint16	AdditionalResourceSubURILengthBytes[PortCount - 1]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[PortCount - 1]	"0/Settings"
uint32	...	...
uint32	AdditionalResourceID [2*PortCount - 2]	210 + PFs[PFCount-1]
uint16	AdditionalResourceSubURILengthBytes[PortCount - 2]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[PortCount - 2]	""n"/Settings"
ver32	MajorSchemaVersion	0xF1F3F000
uint16	MajorSchemaDictionaryLengthBytes	Actual length of dictionary stored in NVM.
uint32	MajorSchemaDictionarySignature	32-bit CRC of the schema dictionary.
unit8	MajorSchemaNameLength	24
strUTF-8	MajorSchemaName	"NetworkDeviceFunction.NetworkDeviceFunction\0"
uint16	OEMCount	0

12.8.3.3.8.7 PCIeFunctions

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	SubURILengthBytes	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

12.8.3.3.8.8 PCIeFunction

Type	Description	Value
uint32	recordHandle	4300
uint32	ResourceID	300 +PFs[0]
bitfield8	ResourceFlags	2 (is_contained_in_collection)
uint32	ContainingResourceID	4030
uint16	ProposedContainingResourceLengthBytes	1
strUTF-8	ProposedContainingResourceName	"\0"
uint16	SubURILengthBytes	Length in bytes of SubURI
strUTF-8	SubURI	"PCIeFunctions/0"
uint16	AdditionalResourceIDCount	PFCount - 1
uint32	AdditionalResourceID [0]	300 +PFs[1]
uint16	AdditionalResourceSubURILengthBytes[0]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[0]	"PCIeFunctions/1"
uint32	...	...
uint32	AdditionalResourceID [PortCount - 2]	300 + PFs[PFCount - 1]
uint16	AdditionalResourceSubURILengthBytes[PortCount - 2]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[PortCount - 2]	"PCIeFunctions/"n""
ver32	MajorSchemaVersion	0xF1F2F100
uint16	MajorSchemaDictionaryLengthBytes	Actual length of dictionary stored in NVM.

Type	Description	Value
uint32	MajorSchemaDictionarySignature	32-bit CRC of the schema dictionary.
uint8	MajorSchemaNameLength	46
strUTF-8	MajorSchemaName	"PCIeFunction.PCIeFunction\0"
uint16	OEMCount	0

12.8.3.3.9 Non-ACD PDRs

There are several schemas that are beyond the ACD profile definition. Figure 12-15 defines those PDRs. The supported fields are based on the OCP Baseline Hardware Management profile v1_0_0.

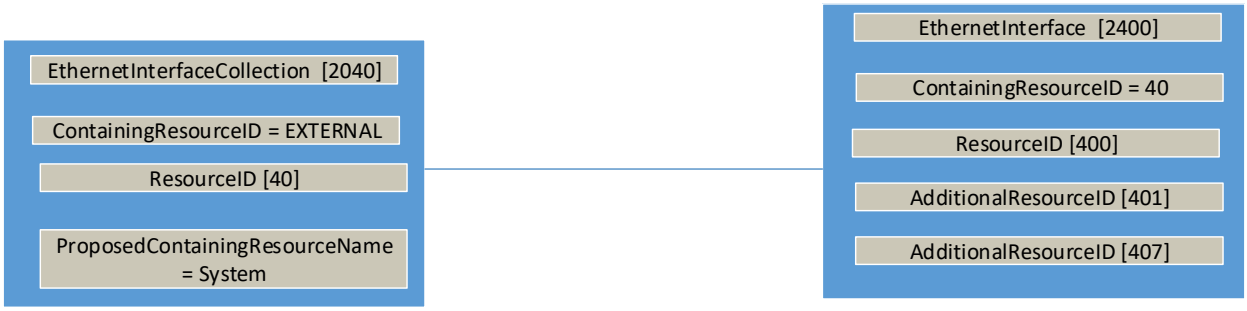


Figure 12-15. Non-ACD PDRs

12.8.3.3.9.1 EthernetInterfaceCollection

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	SubURILengthBytes	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

12.8.3.3.9.2 EthernetInterface

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	SubURILengthBytes	Length in bytes of SubURI
strUTF-8	SubURI	"0"
uint16	AdditionalResourceIDCount	2*PortCount-1
uint32	AdditionalResourceID [0]	2400 + Ports[1]
uint16	AdditionalResourceSubURILengthBytes[0]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[0]	"1"
uint32	...	...
uint32	AdditionalResourceID [PortCount - 2]	400 + Ports[PortCount - 1]
uint16	AdditionalResourceSubURILengthBytes[PortCount - 2]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[PortCount - 2]	"n"
uint32	AdditionalResourceID [PortCount - 1]	410 + Ports[0]
uint16	AdditionalResourceSubURILengthBytes[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	AdditionalResourceSubURILengthBytes[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.
unit8	MajorSchemaNameLength	18
strUTF-8	MajorSchemaName	"EthernetInterface.EthernetInterface"
uint16	OEMCount	0

12.8.3.3.10 RDE Action PDRs

The only action currently supported is the ResetSettingsToDefault action PDR related to NetworkAdapter resource.

See DSP0248 for the PDR definition.

12.8.3.3.10.1 ResetSettingsToDefault

Type	Description	Value
uint8	ActionPDRIndex	0
uint16	RelatedResourceCount	1
uint32	RelatedResourceID [0]	2001
uint8	ActionCount	1
uint8	ActionNameLengthBytes [0]	23
utf8string	ActionName [0]	"ResetSettingsToDefault\0"
uint8	ActionPathLengthBytes [0]	ActionPath length in bytes.
utf8string	ActionPath [0]	Null-terminated string detailing the path to the root of the Action within the resource's major dictionary.

12.8.4 PLDM Firmware Update Commands

The firmware should support the following PLDM commands and responses according to DSP0267.

Table 12-82. PLDM for Firmware Update Command Codes

Command	Command Code	Command Requestor (Initiator)	Supported	Section Reference
QueryDeviceIdentifiers	0x01	BMC	Yes	12.8.4.2.1
GetFirmwareParameters	0x02	BMC	Yes	12.8.4.2.2
RequestUpdate	0x10	BMC	Yes	12.8.4.2.3
GetPackageData	0x11	Firmware	Yes	12.8.4.1.1
GetDeviceMetaData	0x12	BMC	No	---
PassComponentTable	0x13	BMC	Yes	12.8.4.2.4
UpdateComponent	0x14	BMC	Yes	12.8.4.2.5
RequestFirmwareData	0x15	Firmware	Yes	12.8.4.1.2
TransferComplete	0x16	Firmware	Yes	12.8.4.1.3
VerifyComplete	0x17	Firmware	Yes	12.8.4.1.4
ApplyComplete	0x18	Firmware	Yes	12.8.4.1.5
GetMetaData	0x19	Firmware	No	---
ActivateFirmware	0x1A	BMC	Yes	12.8.4.2.6
GetStatus	0x1B	BMC	Yes	12.8.4.2.7
CancelUpdateComponent	0x1C	BMC	Yes	12.8.4.2.8
CancelUpdate	0x1D	BMC	Yes	12.8.4.2.9

12.8.4.1 Firmware-to-BMC Commands

This section describes the commands that are sent by firmware to BMC. Please 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 in case of error or timeout, as explained in "8.2 State machine" in DSP0267.
- In the case of 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) in case of an error in the following states:
    - IDLE
    - LEARN
    - ACTIVATE
  - Cancel Component Update (return to READY XFER):
    - READY XFER
    - DOWNLOAD
    - VERIFY
    - APPLY

12.8.4.1.1 GetPackageData (Command Code 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 6.3.6.2 and Section 6.3.56.

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.

12.8.4.1.2 RequestFirmwareData (Command Code 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.

12.8.4.1.3 TransferComplete (Command Code 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.

12.8.4.1.4 VerifyComplete (Command Code 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 3.4.5.

Firmware should also verify that the parameters received before the download match the downloaded components.

12.8.4.1.5 ApplyComplete (Command Code 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 12.8.5.6.

12.8.4.2 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.

12.8.4.2.1 QueryDeviceIdentifiers (Command Code 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 12.8.5.2.1 for FD *InitialDescriptorType*, *InitialDescriptorLength*, *InitialDescriptorData*.

12.8.4.2.2 GetFirmwareParameters (Command Code 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 RequestUpdate 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 *ComponentImageSetVersionStringType* in Section 12.8.5.2.1.
PendingComponentImageSetVersionStringType	Same as for active set, but for pending component.
ActiveComponentImageSetVersionString	Refer to *ComponentImageSetVersionString* in Section 12.8.5.2.1.
PendingComponentImageSetVersionString	Same format as for active set.
ComponentClassification	Refer to *ComponentClassification* in Section 12.8.5.2.1.
ComponentIdentifier	Refer to *ComponentIdentifier* in Section 12.8.5.2.1.
ComponentClassificationIndex	Not used. Set to 0.
ActiveComponentComparisonStamp	Refer to *ComponentComparisonStamp* in Section 12.8.5.2.1.
ActiveComponentVersionStringType	Refer to *ComponentVersionStringType* in Section 12.8.5.2.1.
ActiveComponentReleaseDate	Refer to *ComponentReleaseDate* in Section 12.8.5.2.1.
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, which includes migrating the new component image to its final non-volatile storage destination.
ActiveComponentVersionString	Refer to *ComponentVersionString* in Section 12.8.5.2.1.
PendingComponentVersionString	Same as for active component.

12.8.4.2.3 RequestUpdate (Command Code 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 ≤ *ComponentCount* in *GetFirmwareParameters*.
PackageDataLength	Should be > 0. See Section 12.8.4.1.1.

Response Parameter	Description
FirmwareDeviceMetaDataLength	Set to 0, as *GetMetaData* is not supported.

12.8.4.2.4 PassComponentTable (Command Code 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 are required to 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 *ComponentClassification* at in Section 12.8.5.2.1.
ComponentIdentifier	Refer to *ComponentIdentifier* in Section 12.8.5.2.1.
ComponentComparisonStamp	Refer to *ComponentComparisonStamp* in Section 12.8.5.2.1.
ComponentVersionStringType	Refer to *ComponentVersionStringType* in Section 12.8.5.2.1.
ComponentVersionString	Refer to *ComponentVersionString* in Section 12.8.5.2.1.

12.8.4.2.5 UpdateComponent (Command Code 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 currently installed. The UA sets this bit for any component that has the force update bit set in the *ComponentOptions* 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.

12.8.4.2.6 ActivateFirmware (Command Code 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.

12.8.4.2.7 GetStatus (Command Code 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% * num_downloaded / total_num_of_comp VERIFY – 100% * num_verified / total_num_of_comp APPLY – 100% * num_applied / total_num_of_comp ACTIVATE – 0%

12.8.4.2.8 CancelUpdateComponent (Command Code 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.

12.8.4.2.9 CancelUpdate (Command Code 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.

12.8.5 PLDM Firmware Update Flow

This section describes the implementation of DSP0267 - PLDM firmware update protocol.

12.8.5.1 Update Components

The firmware update over PLDM can be done to the following NVM components:

- NVM
- OROM
- Netlist

12.8.5.2 Firmware Update Package

This section describes the fields of the Firmware Update Package that are relevant for this implementation.

Note: Support of packages of several devices is done through device ID and already filtered by the BMC, so the FD receives components that are relevant to its device only.

12.8.5.2.1 Package Versions

This section describes the components versions format and identifiers as to be set inside the Firmware Update package. Only the fields that are relevant to this implementation are covered here.

For firmware usage of the fields below, see Section 12.8.4, "PLDM Firmware Update Commands".

The following parameters are to be defined on a package level. See the details in Table 12-83 through Table 12-85.

- Component
  - Version
  - Classification
  - Identifier
  - Comparison Stamp
  - Release Date
- Component Set
  - Version
- Firmware Device
  - Type
  - Identifier

General Versions:

- NVM component version string is an 18-character ASCII-encoded buffer as CCCCCCCC.SSSSSSSS<nul>, where CCCCCCCC and SSSSSSSS are the 32-bit ComponentComparisonStamp and security revision (Iad_srev), respectively, rendered in hex with leading zeros as needed, and terminated by a null byte.
- O-ROM component version string is an 18-character ASCII-encoded buffer as CCCCCCCC.SSSSSSSS<nul>, where CCCCCCCC and SSSSSSSS are the 32-bit ComponentComparisonStamp and security revision (Iad_srev), respectively, rendered in hex with leading zeros as needed, and terminated by a null byte. uEFI OROM version in PCIR structure, as reported by Get Controller Information command (Section 12.6.4.16.1).
- Netlist component version string is:

  Concatenate (BaseReleaseVersion.Major,".",BaseReleaseVersion.Minor,".", BaseReleaseVersion.Type,".",CustomerNetlistVersion.IANA,".",CustomerNetlistVersion)/

  The versions are represented as eight ASCII characters, each with leadings zeros added as needed. The CustomerNetlistVersion is two bytes (four characters) only. The length of this version is fixed to 42 (including null padding).

  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.

- The ComponentSetVersionString for OEM Gen images is encoded as a 32-character string of the form N:nnnnnnnnO:oooooooT:ttttttttt<nul>, where nnnnnnnn, oooooooo, and tttttttt are the 32-bit NVM, OROM, and netlist Component Comparison Stamps, respectively, rendered in hex with leading zeros as needed, and terminated by a null byte.

Table 12-83. 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 14.2.5.2. ***Note:*** All the descriptors field are taken from the config space of PF0.	Yes
uint16	AdditionalDescriptorType1	PCI Subsystem Vendor ID	Yes
Variable	AdditionalDescriptorData1	See Section 14.2.5.10.	Yes
uint16	AdditionalDescriptorType2	PCI Subsystem ID	Yes
Variable	AdditionalDescriptorData2	See Section 14.2.5.11.	Yes
uint16	AdditionalDescriptorType3	PCI Revision ID	No
Variable	AdditionalDescriptorData3	See Section 14.2.5.5.	No

Table 12-84. 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 above for OEM Gen ComponentSetVersionString. When written, the *ComponentImageSetVersionString* is stored in PFV TLV #0x127	Yes

Table 12-85. 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 = 42	Yes
Variable	**ComponentVersionString** Up to 255 bytes	See above.	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 Net List Module ID = 0x8	Yes
uint8	**ComponentClassificationIndex** Used to distinguish identical components that have the same classification and identifier which can use the same component image but the images are stored in different locations in the FD.	Not used.	No
uint32	**ComponentComparisonStamp** When *ComponentOptions* 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 *MajorMinorRevisionPatch* where each subfield has a range of 0x00 to 0xFF.	MajorMinor Major = 16 bit Minor = 16 bit NVM bank comparison stamp = 31:25: DevStarterVersion Major (0-99) 24:18: DevStarterVersion Minor (0-99) 17:0: EETrackID OROM bank comparison stamp = CIVD if Combo - Combo Image Version Low/ High in CIVD Section. In case of no OROM, the version reported is all zeros. Net list comparison stamp = BaseReleaseVersion.Major/Minor (32 bit value copied from Netlist identifier block).	No
ASCII[8]	**ComponentReleaseDate** Optional. Firmware returns this information upon *GetFirmwareParameters* 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

12.8.5.3 Firmware Flow for Section Update

Table 12-86 includes the commands and the firmware action for each command for any bank update.

Note: The PFA section is preserved and not changed by PLDM update.

Table 12-86. 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 9.5.13.6. 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."	12.8.4.2.3
	GetPackageData			12.8.4.1.1
PassComponentTable		Respond success or other error code. Perform the following checking: Check GFID as described in Section 3.4.5.6.1. Check security revision as described in Section 3.4.9.2. If one of the checks fails, return error code 0x07. *ComponentComparisonStamp* >= *ActiveComponentComparisonStamp* as reported by *GetFirmwareParameters*. If not, return error code 0x1- 0x2 accordingly.	Pass component information. Contains the component information table for a specific component including *ComponentClassificationIndex*, *ComponentClassification*, and version details. Number of commands is the number of components to update. Firmware should allow downgrade only up to security version.	12.8.4.2.4
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 3.4.10.2.	12.8.4.2.5
	RequestFirmwareData	Store internally the received data until 4 KB received, then issue an update NVM command. Respond success or other error code.	See Section 3.4.10.3.	12.8.4.1.2
	TransferComplete	Verify the image authentication. The following is according to Section 3.4.5.6 (OROM)/ Section 3.4.5.6 (NVM), Step 6- Step 8. No authentication check is done on Netlist.		12.8.4.1.3

Table 12-86. Action Upon Reception of Commands - NVM Section [continued]

BMC Command	Firmware Device Command	Firmware Device Action	Remarks	Section Reference
	VerifyComplete	Validation checks same as done by NVM Update admin command. Verify that *ComponentVersionString*, *ComponentComparisonStamp*, *ComponentSetVersionString* as defined in (Section 12.8.4.1.1) as passed before the download matches the downloaded NVM/ OROM bank, see in Section 12.8.5.2.1.		12.8.4.1.4
	ApplyComplete			12.8.4.1.5
ActivateFirmware		Update PFA with the following fields according to the TLVs ion *PackageData*. Respond success or other error code.	See Section 3.4.5.2.1 – Results in validate the new bank and invalidate the old bank. See Section 9.5.13.7.	12.8.4.2.6

12.8.5.4 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

12.8.5.5 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 (CORER, GLOBR), the firmware update continue the update.

12.8.5.6 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.

12.8.6 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 12-16. 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.

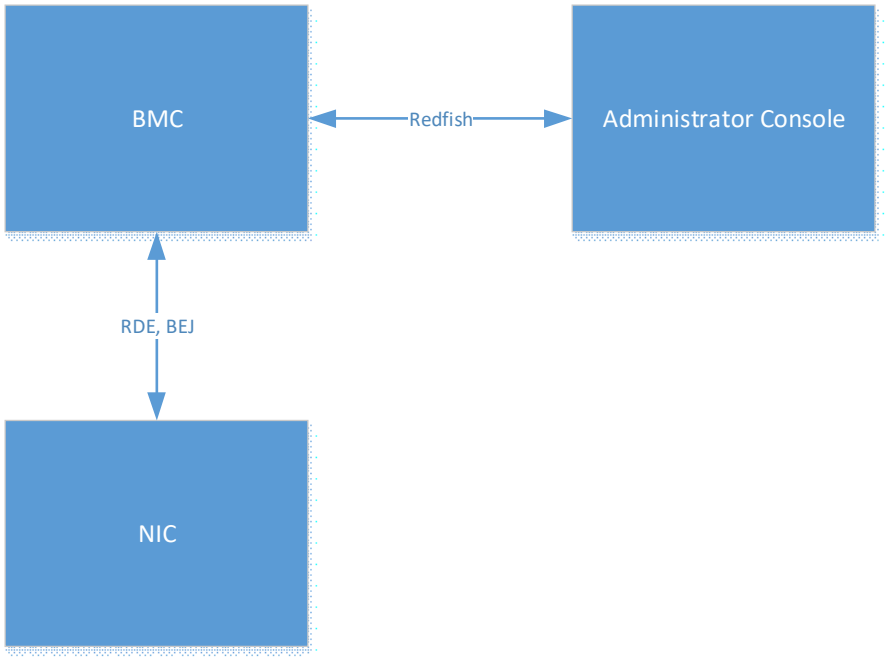


Figure 12-16. RDE Setup

12.8.6.1 Implementation Guidelines

12.8.6.1.1 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

12.8.6.1.2 PUT, PATCH Operation Guidelines

If an operation of type UPDATE or REPLACE is received with a BEJ payload containing a mix of Read-Only and Read-Write properties, the operation silently ignores all Read-Only properties and behaves as if the payload contains Read-Write properties only.

If the payload of a write operation contains Read-Only properties exclusively, the operation ignores the entire payload, performs no write, and returns success.

12.8.6.1.3 String Handling

Any string received with a null character in the middle of the string is treated as an invalid input.

12.8.6.1.4 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 12-90, "Resources Parameters" on page 1772 for a list of allowed operation per resource. Permissions on a Property level are not individually supported, and all errors are silently ignored.

12.8.6.1.5 Resources Exposing @Redfish.Settings Attr.

For the NetworkPort, NetworkDeviceFunction, and EthernetInterface schemas, there are two sets of resources, as follows:

Schema	Regular Resource ID (RO)	Settings Resource ID (RW)
NetworkPort	100-107	110-117
NetworkDeviceFunction	200-207	210-217
EthernetInterface	400-407	410-417

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 (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).

12.8.6.2 RDE Commands Summary

All the RDE commands should be implemented according to DSP0218 and follow the limitations/ guidelines as defined in this chapter.

Table 12-87. RDE Commands Summary Table

Command	Command Code	Command Requirement for RDE Device	Section Reference
Discovery and Schema Management Commands			
NegotiateRedfishParameters	0x01	Supported	12.8.6.3.1
NegotiateMediumParameters	0x02	Supported	12.8.6.3.2
GetSchemaDictionary	0x03	Supported	12.8.6.3.3
GetSchemaURI	0x04	Supported	12.8.6.3.4
GetResourceETag	0x05	Supported	12.8.6.3.5
Reserved	0x06 - 0x0F	---	---
RDE Operation and Task Commands			
RDEOperationInit	0x10	Supported	12.8.6.3.6
SupplyCustomRequestParameters	0x11	Supported	12.8.6.3.7
RetrieveCustomResponseParameters	0x12	Not Supported	12.8.6.3.8
RDEOperationComplete	0x13	Supported	12.8.6.3.9
RDEOperationStatus	0x14	Supported	12.8.6.3.10
RDEOperationKill	0x15	Supported	12.8.6.3.11
RDEOperationEnumerate	0x16	Supported	12.8.6.3.12
Reserved	0x17 - 0x2F	---	---
Multi-part Transfer Commands			
MultipartSend	0x30	Supported	12.8.6.3.13
MultipartReceive	0x31	Supported	12.8.6.3.14
Reserved	0x32 - 0x3F	---	---
Reserved			
Reserved	0x40 - 0xFF	---	---

12.8.6.3 Command Details

12.8.6.3.1 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 *MCConcurrencySupport* is equal to zero, return a completion code of ERROR_INVALID_DATA.
bitfield16	**MCFeatureSupport** Firmware ignores this field.
Type	**Response Data**
enum8	**CompletionCode** Value: {PLDM_BASE_CODES}
uint8	**DeviceConcurrencySupport** Return 1.
bitfield8	**DeviceCapabilitiesFlags** Capabilities for this RDE device. [7:2] = Reserved [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 ALGORITHM_MODE = SHA256. 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 *DeviceProviderName*. Otherwise, use “Name” property at NetworkInterface.

12.8.6.3.2 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.

12.8.6.3.3 GetSchemaDictionary (0x03)

Type	Request Data
uint32	**ResourceID** The *ResourceID* of any resource in the Redfish Resource PDR from which to retrieve the dictionary. See Table 12-88 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 12-88 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 *RequestedSchemaClass* 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 *CompletionCode*, the RDE device returns a valid transfer handle as described in Table 12-88.

Table 12-88. Handle per Resource

Schema (*.json)	Dictionary Class Supported	Resource ID
Event.v1_3_1	EVENT	0xFFFF FFFF or any of the other Resource IDs below.
redfish-payload-annotations.v1_0_0	ANNOTATION	
redfish-error.v1_0_0	ERROR	
NetworkAdapter.v1_2_0	MAJOR	1
PCIeDevice.v1_3_0	MAJOR	3
NetworkInterface.v1_1_1	MAJOR	5
NetworkPortCollection	MAJOR	10
	COLLECTION_MEMBER_TYPE	10
PCIeFunctionCollection	MAJOR	30
NetworkDeviceFunctionCollection	MAJOR	20
NetworkDeviceFunction.v1_3_0	COLLECTION_MEMBER_TYPE	20
NetworkPort.v1_2_1	MAJOR	100-107 110-117
NetworkDeviceFunction.v1_3_0	MAJOR	200-207 210-217
PCIFunction.v1_2_1	MAJOR	300-307
EthernetInterface.v1_5_1	MAJOR	400-407 410-417
EthernetInterfaceCollection	MAJOR	40
	COLLECTION_MEMBER_TYPE	40

12.8.6.3.4 GetSchemaURI (0x04)

Type	Request Data
uint32	**ResourceID** The *ResourceID* of any resource in the Redfish Resource PDR from which to retrieve the URI. See Table 12-89 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:
	{
	PLDM_BASE_CODES,
	ERROR_UNSUPPORTED,
	ERROR_NO_SUCH_RESOURCE
	}

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 12-89. 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	https://redfish.dmtf.org/schemas/NetworkAdapter.v1_3_0.json
NetworkInterface	MAJOR	5	https://redfish.dmtf.org/schemas/NetworkInterface.v1_1_3.json
PCIeDevice	MAJOR	3	https://redfish.dmtf.org/schemas/PCIeDevice.v1_4_0.json (not implemented)
NetworkPortCollection	COLLECTION_MEMBER_TYPE	10	https://redfish.dmtf.org/schemas/NetworkPort.json
	MAJOR		https://redfish.dmtf.org/schemas/NetworkPortCollection.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
NetworkPort	MAJOR	100-107 110-117	https://redfish.dmtf.org/schemas/NetworkPort.v1_2_3.json
NetworkDeviceFunction	MAJOR	200-207 210-217	https://redfish.dmtf.org/schemas/NetworkDeviceFunction.v1_3_3.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

12.8.6.3.5 GetResourceETag (0x05)

Type	Request Data
uint32	**ResourceID** The *ResourceID* 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. For non-existent *ResourceID*, ERROR_NO_SUCH_RESOURCE *CompletionCode* 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 *CompletionCode* is not SUCCESS. Firmware should calculate the data that would have been returned for the READ operation of the *ResourceID* and calculate the ETag for it. It should be the same *ETAG* value as returned as part of a read completion of *ResourceID*. Use the same algorithm as for signature at GetSchemaDictionary. ETag is calculated only on the data that is immediately contained within the resource.

12.8.6.3.6 RDEOperationInit (0x10)

Type	Request Data
uint32	**ResourceID** The *ResourceID* of a resource in the Redfish Resource PDR for the data that is the target of this operation. The supported *ResourceIDs* are listed in Table 12-90.
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 12-90.
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 *OperationLocator* field is ignore
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 is0x00000000 (see the *RequestPayloadLength* and *RequestPayload* fields below).
uint8	**OperationLocatorLength** Length in bytes of the *OperationLocator* for this Operation. This field is zero if the *locator_valid* bit in the *OperationFlags* 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 *contains_request_payload* bit of the *OperationFlags* 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 *ResourceID*. Supported for Read, Update, and Action operations. This field is omitted if the *OperationLocatorLength* 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 *RequestPayloadLength* 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 *OperationFlags.contains_custom_request_parameters*=1 OR *OperationFlags.contains_request_payload*=1 AND *SendDataTransferHandle* <> 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 12.8.6.5.2 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 *OperationType* 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. Bits 0 and 5 are always set. Bits 1 and 2 are set if update and replace operation are allowed. Bits 3 and 4 are always cleared If CompletionCode = ERROR_NOT_ALLOWED, this field returns 0x00.
uint32	**ResponsePayloadLength** As described in specification. Currently no *@Message.ExtendedInfo* 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 *ResponsePayloadLength* 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 12-90. Resources Parameters

Schema	Resource ID	Allowed Operations
NetworkAdapter	1	Read, Head, Update, Replace, Action
NetworkInterface	5	Read, Head, Update, Replace
PCIeDevice	3	Read, Head, Update, Replace (not implemented)
NetworkPortCollection	10	Read, Head
NetworkDeviceFunctionCollection	20	Read, Head
NetworkPort	100-107	Read, Head
NetworkDeviceFunction	200-207	Read, Head
PCIFunction	300-307	Read, Head, Update, Replace
EthernetInterface	400-407	Read, Head
EthernetInterfaceCollection	40	Read, Head
Setting Resources	110-117 210-217 410-417	Read, Head, Update, Replace

12.8.6.3.7 SupplyCustomRequestParameters (0x11)

Type	Request Data
uint32	**ResourceID** The *ResourceID* of a resource in the Redfish Resource PDR for the data that is the target of this operation. The supported *ResourceIDs* are listed in Table 12-90, "Resources Parameters".
rdeOpID	**OperationID** The {*ResourceID*, *OperationID*} 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 *ETagOperation* = ETAG_IF_MATCH = 1, do the action only if the calculated ETag for the resource == ETag [0]. If *ETagOperation* = ETAG_IF_MATCH and EtagCount <>1, return a ERROR_INVALID_DATA error. If *ETagOperation* = ETAG_IF_NONE_MATCH = 2, do the action only if the calculated ETag for the resource <> all of {ETag [0] … ETag[*ETagCount*-1]. If *ETagOperation* = ETAG_IF_NONE_MATCH and *EtagCount* == 0, return a ERROR_INVALID_DATA error. If *ETagOperation* = ETAG_IGNORE = 0 and *EtagCount* <> 0 return a ERROR_INVALID_DATA error If *ETagOperation* >= 3, return a ERROR_INVALID_DATA error.
uint8	**ETagCount** Number of ETags supplied in this message. Should be zero if *ETagOperation* above is ETAG_IGNORE, and nonzero otherwise.
varstring	**ETag [0]** String data for first ETag, if *ETagCount* > 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:
	{
	PLDM_BASE_CODES,
	ERROR_ OPERATION_ABANDONED,
	ERROR_ OPERATION_FAILED,
	ERROR_UNSUPPORTED,
	ERROR_UNEXPECTED,
	ERROR_UNRECOGNIZED_CUSTOM_HEADER,
	ERROR_ETAG_MATCH,
	ERROR_NO_SUCH_RESOURCE
	}
	When an unknown *ResourceID* is supplied, ERROR_NO_SUCH_RESOURCE is returned.
	If *OperationID* references an unknown operation, or *OperationID* and *ResourceID* 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:
	{
	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
	}

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 *OperationExecutionFlags* in Section 12.8.6.3.6, "RDEOperationInit (0x10)".
uint32	**ResultTransferHandle** See *ResultTransferHandle* in Section 12.8.6.3.6, "RDEOperationInit (0x10)".
bitfield8	**PermissionFlags** See *PermissionFlags* in Section 12.8.6.3.6, "RDEOperationInit (0x10)".
uint32	**ResponsePayloadLength** See *ResponsePayloadLength* in Section 12.8.6.3.6, "RDEOperationInit (0x10)".
varstring	**ETag** See *ETag* in Section 12.8.6.3.6, "RDEOperationInit (0x10)".
null or bejEncoding	**ResponsePayload** See *ResponsePayload* in Section 12.8.6.3.6, "RDEOperationInit (0x10)".

12.8.6.3.8 RetrieveCustomResponseParameters (0x12)

This command is currently not supported, as the NIC does not provide any Custom Response parameters.

Type	Request Data
uint32	**ResourceID** The *ResourceID* of a resource in the Redfish Resource PDR for the data that is the target of this operation. he supported *ResourceIDs* are listed in Table 12-90, "Resources Parameters".
rdeOpID	**OperationID** The {*ResourceID*, *OperationID*} 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 *ResourceID* is supplied, ERROR_NO_SUCH_RESOURCE is returned.
	If *OperationID* references an unknown operation, or *OperationID* and *ResourceID* 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
...	...

12.8.6.3.9 RDEOperationComplete (0x13)

Type	Request Data
uint32	**ResourceID** The *ResourceID* of a resource in the Redfish Resource PDR for the data that is the target of this operation. The supported *ResourceIDs* are listed in Table 12-90, "Resources Parameters".
rdeOpID	**OperationID** The {*ResourceID*, *OperationID*} 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 *ResourceID* is supplied, ERROR_NO_SUCH_RESOURCE is returned. If *OperationID* references an unknown operation, or *OperationID* and *ResourceID* do not both match an existing operation, ERROR_UNEXPECTED is returned.

12.8.6.3.10 RDEOperationStatus (0x14)

Returns status of the current task.

Type	Request Data
uint32	**ResourceID** The *ResourceID* of a resource in the Redfish Resource PDR for the data that is the target of this operation. The supported *ResourceIDs* are listed in Table 12-90, "Resources Parameters".
rdeOpID	**OperationID** The {*ResourceID*, *OperationID*} 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, E
	RROR_ETAG_MATCH, E
	RROR_UNRECOGNIZED_
	CUSTOM_HEADER
	}
	When an unknown *ResourceID* is supplied ERROR_NO_SUCH_RESOURCE is returned.
	If *OperationID* references an unknown operation, or *OperationID* and *ResourceID* 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 *OperationID* request field and it is not in the failed state. The actual current status of the RDE Operation is returned in the *OperationStatus* field - OR - an inactive RDE Operation was referenced in the *OperationID* request field. *OperationStatus* is OPERATION_INACTIVE in this case.
	ERROR_UNSUPPORTED, ERROR_ETAG_MATCH, ERROR_UNRECOGNIZED_CUSTOM_HEADER: A valid active RDE Operation was referenced in the *OperationID* request field, but the Operation failed with the specified status code. *OperationStatus* 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 *OperationExecutionFlags* in Section 12.8.6.3.6, "RDEOperationInit (0x10)".
uint32	**ResultTransferHandle** See *ResultTransferHandle* in Section 12.8.6.3.6, "RDEOperationInit (0x10)".
bitfield8	**PermissionFlags** See *PermissionFlags* in Section 12.8.6.3.6, "RDEOperationInit (0x10)".
uint32	**ResponsePayloadLength** See *ResponsePayloadLength* in Section 12.8.6.3.6, "RDEOperationInit (0x10)".
varstring	**ETag** See *ETag* in Section 12.8.6.3.6, "RDEOperationInit (0x10)".
null or bejEncoding	**ResponsePayload** See *ResponsePayload* in Section 12.8.6.3.6, "RDEOperationInit (0x10)".

12.8.6.3.11 RDEOperationKill (0x15)

Type	Request Data
uint32	**ResourceID** The *ResourceID* of a resource in the Redfish Resource PDR for the data that is the target of this operation. The supported *ResourceIDs* are listed in Table 12-90, "Resources Parameters".
rdeOpID	**OperationID** The {*ResourceID*, *OperationID*} should match the value set in the RDEOperationInit command. Otherwise, an ERROR_UNEXPECTED is returned.
bitfield8	**KillFlags** Flags for killing the Operation: [7:2] = Reserved for future use [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 no 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: {PLDM_BASE_CODES, ERROR_OPERATION_ABANDONED, ERROR_OPERATION_FAILED, ERROR_OPERATION_UNKILLABLE, ERROR_NO_SUCH_RESOURCE, ERROR_UNEXPECTED } When an unknown *ResourceID* is supplied, ERROR_NO_SUCH_RESOURCE is returned. If *OperationID* references an unknown operation, or *OperationID* and *ResourceID* do not both match an existing operation, ERROR_UNEXPECTED is returned. If *KillFlags.run_to_completion* is set, but *KillFlags.discard_record* is cleared, return ERROR_INVALID_DATA. If both *KillFlags.run_to_completion* and *KillFlags.discard_record* 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.

12.8.6.3.12 RDEOperationEnumerate (0x16)

Type	Request Data
N/A	This request contains no parameters.
Type	Response data.
enum8	**CompletionCode** Value: {PLDM_BASE_CODES}
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: {OPERATION_HEAD = 0, OPERATION_READ = 1, OPERATION_UPDATE = 4, OPERATION_REPLACE = 5, OPERATION_ACTION = 6 } This field is omitted if *OperationCount* above is zero.

12.8.6.3.13 MultipartSend (0x30)

Type	Request Data
uint32	**DataTransferHandle** A handle to uniquely identify the chunk of data to be sent. If *TransferFlag* below is START or START_AND_END, this must match the *SendDataTransferHandle* that was supplied by the RDE Device in the response to RDEOperationInit. The *DataTransferHandle* supplied is either the initial handle to begin or restart a transfer, or the *NextDataTransferHandle* as specified in the previous chunk.
rdeOpID	**OperationID** The *OperationID* 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:
	{
	START = 0,
	MIDDLE = 1, E
	ND = 2,
	START_AND_END = 3
	}

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 *Data* field is omitted from the request message if the value of *DataLengthBytes* 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 (TransferFlag ≠ 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: {PLDM_BASE_CODES, ERROR_OPERATION_ABANDONED, ERROR_OPERATION_FAILED, ERROR_UNEXPECTED, ERROR_BAD_CHECKSUM } If *OperationID* is non-zero and references an unknown operation ERROR_UNEXPECTED is returned. If the *DataTransferHandle* does not correspond to a valid chunk, the RDE Device returns *CompletionCode* 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:
	{
	XFER_FIRST_PART = 0,
	XFER_NEXT_PART = 1,
	XFER_ABORT = 2,
	XFER_COMPLETE = 3
	}

12.8.6.3.14 MultipartReceive (0x31)

Type	Request Data
uint32	**DataTransferHandle** If *TransferOperation* below is XFER_FIRST_PART and the *OperationID* below is zero, this must match the *TransferHandle* supplied by the RDE Device in the response to the GetSchemaDictionary command. If *TransferOperation* below is XFER_FIRST_PART and the *OperationID* below is nonzero, this must match the *SendDataTransferHandle* that was supplied by the RDE Device in the response to RDEOperationInit. If *TransferOperation* below is XFER_NEXT_PART, this must match the *NextDataHandle* supplied by the RDE Device with the previous chunk. The *DataTransferHandle* supplied is either the initial handle to begin or restart a transfer or the *NextDataTransferHandle* supplied with the previous chunk.
rdeOpID	**OperationID** The *OperationID* 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:
	{
	XFER_FIRST_PART = 0,
	XFER_NEXT_PART = 1,
	XFER_ABORT = 2
	}

Type	**Response Data**

enum8	**CompletionCode**
	Values:
	{
	PLDM_BASE_CODES,
	ERROR_OPERATION_ABANDONED,
	ERROR_OPERATION_FAILED,
	ERROR_UNEXPECTED,
	ERROR_BAD_CHECKSUM
	}
	If *OperationID* is non-zero and references an unknown operation, ERROR_UNEXPECTED is returned.
	If the *DataTransferHandle* does not correspond to a valid chunk, the RDE Device returns *CompletionCode* ERROR_INVALID_DATA.
	If the transfer is aborted, the RDE device acknowledges this status by returning SUCCESS.

enum8	**TransferFlag**
	Values:
	{
	START = 0,
	MIDDLE = 1,
	END = 2, S
	TART_AND_END = 3
	}
	This field is omitted for a non-SUCCESS *CompletionCode* 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 *CompletionCode* 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 *CompletionCode* or if the transfer has been aborted.
uint8	**Data [0]** The first byte of current chunk of data. The *Data* field is omitted from the response message if the value of *DataLengthBytes* above is zero This field is omitted for a non-SUCCESS *CompletionCode* 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 *CompletionCode* 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 FW.

12.8.6.4 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 12-91.

Table 12-91. 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 12-17. 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*.

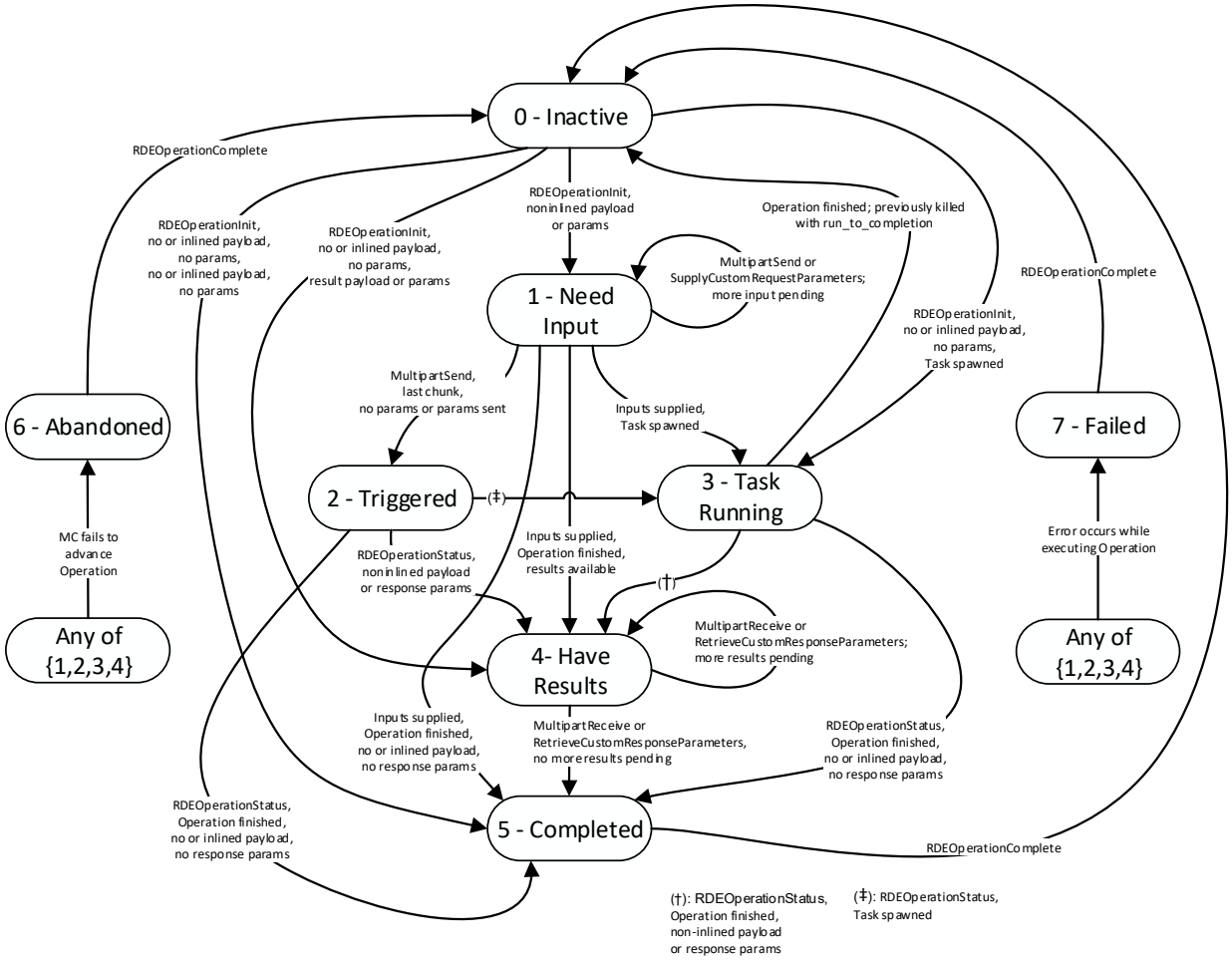


Figure 12-17. Operation Life Cycle State Machine (RDE Device Perspective)

12.8.6.5 Schemas

The following schemas can be found in DSP8010 at https://www.dmtf.org/standards/redfish

This chapter presents the schemas and specifies the relevant properties and where in hardware they refer.

Properties or options that are not mentioned here can be considered as “not supported”.

12.8.6.5.1 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 "E810" 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.

12.8.6.5.2 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 NetworkPortCollection, the @odata_type is:
	`#NetworkPortCollection.NetworkPortCollection`
		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, *sensorID* = 5). Normal = "OK" Upper Non-Critical = "Warning" Upper Critical = "Critical" Upper Fatal = "Critical" If the Card Composite State Sensor (*sensorID* = 5) is uninitialized, Network Controller State sensor (*sensorID* = 50) shall be used instead. If this sensor too 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.

12.8.6.5.3 ACD Schemas

12.8.6.5.3.1 NetworkPort v1.2.3

Note: Function-to-port mapping is linear. Port 0 represents the port that is mapped to function 0.

Note: Resource IDs 100-107 are RO. All changes are done through resource IDs 110-117. After reset (PCIR or EMPR), the contents of 110-117 is applied and copied to 100-107.

Property	Source	Cacheable	Writable	Notes
ActiveLinkTechnology	0 (Ethernet)	Yes	No	
AssociatedNetworkAddresses	NVM, List of LAN and PF MACs	Yes	No	
CurrentLinkSpeedMbps	Use FW API: Link_get_speed Persistent location in NVM: Set the matching PHY types in TLV 0x134.	No	Yes	Integer: {1000, 10000, 25000, 40000, 50000, 100000}
EEEEnabled	Use FW API: PowerDrv_get_EEER Persistent location in NVM: EEE Override Enable field in Link Default Override Mask PFA TLV. Currently hard coded to FALSE. May be set to TRUE if a BASE-T PHY supporting EEE is connected.	No	Yes	Bool: {TRUE, FALSE}
FlowControlConfiguration	Use FW API: Link_get_negotiated_LFC Persistent location in NVM: Set the Link FC options in TLV 0x134.	No	Yes	Enum: {None, TX, RX, TX_RX}
FlowControlStatus	Use FW API: Link_get_negotiated_LFC	No	No	Enum: {None, TX, RX, TX_RX}
LinkStatus	Use FW API: Link is up	No	No	Enum: {Down, Up}
MaxFrameSize		Yes	No	
Name	"E810 Network Port Current Settings" for ResourceID 100-107. "E810 Network Port Pending Settings" for ResourceID 110-117.	Yes	No	
NetDevFuncMaxBWAlloc		Yes	No	
MaxBWAllocPercent	100%	Yes	No	
NetDevFuncMinBWAlloc		Yes	No	
MinBWAllocPercent	100%	Yes	No	
Oem	N/A			
PhysicalPortNumber	Port number (FW)	Yes	No	
PortMaximumMTU	MaxFrameSize - 18	Yes	No	
SignalDetected	DNL Set Link Up, Signal Detect field	No	No	Link API needed
Status	"Enabled" or "Disabled" based on PRTGEN_STATUS.*PORT_VALID* value.	No	No	
SupportedEthernetCapabilities	{None, EEE, WakeOnLAN}. EEE - Either EEE LPI or EEE LLDP enabled, based on EMP Capability bit, PHY Capability, NVM EMP settings. WoL - PFPM_APM.APME	Yes	No	
SupportedLinkCapabilities				

Property	Source	Cacheable	Writable	Notes
AutoSpeedNegotiation	Indicates whether the port has the capability to auto negotiate speed.	Yes	No	Bool
CapableLinkSpeedMbps	Via FW API: Use the list from Notes.	Yes	No	Integer: {1000, 10000, 25000, 40000, 50000, 100000}
LinkNetworkTechnology	0 (Ethernet)	Yes	No	
VendorId	GLPCI_VENDORID.*VENDORID* (presented in decimal format)	Yes	No	
WakeOnLANEnabled	WoL Control per port, NVM EMP settings. Persistent location in NVM: Pointer to PFPM_APM offset - 0x0002	Yes	Yes	
@redfish.Settings				Reported only in resources 100-107.
SettingsObject	Points to the next setting = Resource ID +10	No	No	
SupportedApplyTimes	["OnReset"]	No	No	
ID	Resource Offset	Yes	No	

12.8.6.5.3.2 NetworkPortCollection

Property	Source	Cacheable	Writable	Notes
Name	"NetworkPorts"	Yes	No	
Members@odata.count	Size of Members array.	Yes	No	
Members	Resource IDs of all the exposed NetworkPort resources.	Yes	No	100-107

12.8.6.5.3.3 NetworkInterface v1.1.3

Property	Source	Cacheable	Writable	Notes
Links.NetworkAdapter	NetworkAdapter Resource ID.		No	
Name	"E810 Network Interface"	Yes	No	
NetworkDeviceFunctions	NetworkDeviceFunctionCollection	Yes	No	
NetworkPorts	NetworkPortCollection	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 FW update is in progress. "Enabled" - Otherwise.	No	No	
ID	"%I5"	Yes	No	Deferred binding flag must be set.

12.8.6.5.3.4 NetworkAdapter v1.3.0

Property	Source	Cacheable	Writable	Notes
Actions	ResetSettingsToDefault			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).
Assembly	N/A	Yes	No	Not supported.
Controllers				
ControllerCapabilities				
DataCenterBridging				
Capable	TRUE	Yes	No	
NPAR	N/A			
NPIV	N/A			
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	Array of NetworkPort references.	Yes	No	
PCIeDevices	Array of PCIeDevices references.	Yes	No	No implemented.
PCIeInterface				
LanesInUse	Read from PCIe Configuration space at Link Status Register at Negotiated Link Width.	Yes	No	
MaxLanes	16 = 25x25 mm package 8 = 21x21mm package Read from GL_UFUSE_SOC[7]. 0 = 16 1 =8	Yes	No	
MaxPCIeType	Config space of function 0 - Link capabilities register (0xAC) bits 3:0 - Maximum Link Speed.	Yes	No	

Property	Source	Cacheable	Writable	Notes
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	"E810-CAM2", "E810-CAM1", "E810-XXVAM2"	Yes	No	Read default device ID from PFPCI_DEVID.PF_DEV_ID; xxAM2 from GL_UFUSE_SOC[3:2]
Name	"E810 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 FW update is in progress. "Enabled" - Otherwise.	No	No	Same as in NetworkInterface schema.
Idnetifiers	N/A			
ID	"%11"	Yes	No	Deferred binding flag must be set.

12.8.6.5.3.5 NetworkDeviceFunction v1.3.3

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.

Property	Source	Cacheable	Writable	Notes
AssignablePhysicalPorts	Resource IDs of the ports available in this SKU (100-107).	Yes	No	
AssignablePhysicalPorts@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	

Property	Source	Cacheable	Writable	Notes
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.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	
Status.State	"StandbyOffline" - If in D3/DR state. "Starting" - If in D0 state and host driver is not loaded. "Updating" - If NVM FW update is in progress. "Enabled" - 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	

12.8.6.5.3.6 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

12.8.6.5.3.7 PCIeDevice.v1.4.0

Note: This schema is not implemented, as no way to tie it to the global schemas. Will be implemented later when DSP0218 adds this capability.

Property	Source	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	
Model	"E810"	Yes	No	
Name	"E810"	Yes	No	
PCIeInterface.LanesInUse	Config space of function 0 - Link Status Register (0xB2) bits 9:4 - *Negotiated Link Width*	No	No	
PCIeInterface.MaxLanes	Config space of function 0 - Link Capabilities Register (0xAC) bits 9:4 - *Maximum Link Width*	Yes	No	
PCIeInterface.MaxPCIeType	Config space of function 0 - Link Capabilities Register (0xAC) bits 3:0 - *Maximum Link Speed*	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 - *Current Link Speed*	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

Property	Source	Cacheable	Writable	Notes
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 FW update is in progress. "Enabled" - Otherwise.	No	No	
PCIeFunctions	Link to PCIeFunctionCollection.	Yes	No	
ID				

12.8.6.5.3.8 PCIeFunctionCollection

Property	Source	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

12.8.6.5.3.9 PCIeFunction.v1.2.3

Note: Only Physical functions are exposed through RDE.

Property	Source	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	
Name	"E810"	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 FW 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.*VENDOR_D* (presented in hexadecimal format)	Yes	No	
ID	Resource Offset	Yes	No	

12.8.6.5.4 Non-ACD Schemas

12.8.6.5.4.1 EthernetInterface v1.5.1

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	“E810 Ethernet Interface Current Settings” for ResourceID 400-407. “E810 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	

12.8.6.5.4.2 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

12.8.6.5.5 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 6.3.58.10).

12.8.6.5.6 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.

12.8.6.6 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 may 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.

12.8.6.6.1 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 12-92. 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.

12.9 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 12.6.4.12).

If TCO Isolate is enabled in the NVM (Section 6.3.57.3), the TCO Isolate command disables PCIe write operations to the LAN port. As the software device driver needs to access the CSR space to provide descriptors to the NIC, this operation also stops the network traffic including OS2BMC and MC-to-OS traffic as soon as the existing transmit and receive descriptor queues are exhausted.

MCTP over PCIe VDM are still available in this mode.

12.10 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 E810:

Table 12-93. OCP NIC 3.0 Support

Feature	Notes
NC-SI 1.1	See Section 12.6.
MCTP 1.3	See Section 12.8.
MAC Address Provisioning	Requires NC-SI 1.2. Assumes current draft. May change.
Temperature Reporting - ASIC	See Section 12.7.4.1.
Temperature Reporting - Modules	See Section 12.7.4.1.
SFP/QSFP Modules Power Reporting	See Section 12.7.4.1.
Board Power Reporting	FRU over I^2C. Defined at the board level.
Firmware Inventory and Update	See Section 12.8.4.
Secure Firmware	See Section 3.4.9.
NC-SI Package Addressing	See Section 12.3.2.2.1.
FAN_ON_AUX Pin	See Section 12.10.1.

12.10.1 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.

NOTE: ***This page intentionally left blank.***

Chapter 13 Programming Interface

13.1 Introduction

This section details the programmer visible state inside the E810. In some cases, it describes hardware structures invisible to software in order to clarify a concept.

The E810 address space is mapped into four regions with PCI Base Address registers described in Section 14.2.6.1. These regions are listed in Table 13-1.

Table 13-1. Address Space Regions

Addressable Content	Mapping Style	Region Size
Memory BAR (Internal registers, memories and Flash)	Direct memory-mapped	4 MB - 128 MB
I/O BAR (optional Internal registers)	I/O Window mapped	32 bytes[1]
MSI-X BAR (optional)	Direct memory-mapped	64 KB
Expansion ROM BAR (optional)	Direct memory-mapped	64 KB - 8 MB

1. The internal registers can be accessed though I/O space indirectly, as explained in the sections that follow.

Rules for unsupported accesses:

- Accesses to non-implemented or disabled regions within a BAR are dropped for write accesses, or respond with arbitrary data for read accesses. A PCIe error event is not generated and completions return with successful status.
- Section 3.1.2.2 describes supported PCIe access sizes to each of the BARs and its components.

13.1.1 Access Mechanisms

13.1.1.1 Memory-Mapped Access to Internal Registers and Memories

The internal registers and memories might be accessed as direct memory-mapped offsets from the Base Address register (BAR0 or BAR 0/1 see Section 14.2.6.1).

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.

13.1.1.2 Memory-Mapped Accesses to Flash

The external Flash can be accessed using direct memory-mapped offsets from the memory base address register (BAR0 in 32-bit addressing or BAR0/BAR1 in 64-bit addressing; see Section 14.2.6.1). See Table 13-4 for the location of Flash memory within the memory BAR. Access to Flash memory is restricted to the first 64 KB when GLPCI_LBARCTRL.*FLASH_EXPOSE* is set to 0b.

See Section 3.4 for details on accessing the NVM.

13.1.1.3 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 or BAR3/4; see Section 14.2.6.1). See Section 14.3.3 for the appropriate offset for each specific internal MSI-X register.

In IOV mode, this area is duplicated per VF. It requires a memory space of the maximum between 16 KB and the page size.

13.1.1.4 Memory-Mapped Access to Expansion ROM

The external Flash can also be accessed as a memory-mapped Expansion ROM. Accesses to offsets starting from the Expansion ROM base address (see Section 14.2.6.2) reference the Flash provided that access is enabled from NVM, and if the Expansion ROM base address register contains a valid (non-zero) base memory address.

13.1.1.5 I/O-Mapped Access to Internal Registers

To support pre-boot operation, all internal registers in the regular CSR space can be accessed using I/O operations. I/O accesses are supported only if an I/O base address is allocated and mapped (BAR2; see Section 14.2.6.1), 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, and then the IODATA register is used as a window to the register address specified by IOADDR, as listed in Table 13-2.

Table 13-2. IOADDR and IODATA in I/O Address Space

Offset	Abbreviation	Name	RW	Size
0x00	IOADDR	Internal Register Address. Covers the 8 MB CSR space. 0x00000-0x7FFFFF – Internal Registers. 0x3F0000-0xFFFFFFFF – Undefined.	RW	4 bytes
0x04	IODATA	Data field for reads or writes to the Internal Register location as identified by the current value in IOADDR. All 32 bits of this register can be read from and written to.	RW	4 bytes
0x08-0x1F	Reserved	Reserved.	RO	4 bytes

13.1.1.5.1 IOADDR (I/O Offset 0x00)

The IOADDR register must always be written as a DWord access. Section 3.1.2.3 describes how other access sizes are handled.

Note: 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.

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.

13.1.1.5.2 IODATA (I/O Offset 0x04)

The IODATA register must always be written as a DWord access (assuming the IOADDR register contains a value for the internal register space). Reads to IODATA returns a DWord of data. Section 3.1.2.3 describes how other access sizes are handled.

Notes:

- For software programmers, the IN and OUT instructions must be used to cause I/O cycles to be used on the PCIe bus. Where 32-bit quantities are required on writes, the source register of the OUT instruction must be EAX (the only 32-bit register supported by the OUT command).
- 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 E810 delays the results through normal bus methods (for example, split transaction or transaction retry).
- Because a register 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.

13.1.1.5.3 Undefined I/O Offsets

I/O offsets 0x08 through 0x1F are considered to be reserved offsets with the I/O window. Reads from these addresses return 0xFFFF.

13.1.1.6 Configuration Access to Internal Registers

To support legacy pre-boot 16-bit operating environments without requiring I/O address space, the E810 enables accessing CSRs via the configuration address space by mapping IOADDR and IODATA registers into the configuration address space. If the GLPCI_CAPSUP.*CSR_CONF_EN* bit is set to 1b, access to CSRs via configuration address space is enabled.The register mapping in this case is listed in Table 13-3.

Table 13-3. IOADDR and IODATA in Configuration Address Space

Configuration Address	Abbreviation	Name	RW	Size
0x98	IOADDR	Internal Register Address. Covers the 8 MB CSR space. 0x00000-0x7FFFFF – Internal Registers. 0x3F0000-0xFFFFFFFF – Undefined.	RW	4 bytes
0x9C	IODATA	Data field for reads or writes to the internal register location as identified by the current value in IOADDR. All 32 bits of this register can be read from and 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 should be set to 1b.
   - Bits 30:0 of IOADDR should hold the actual address of the internal register being written to.
2. Data to be written is written into IODATA.
   - IODATA is used as a window to the register address specified by IOADDR. As a result, the data written to IODATA is written into the CSR pointed to by Bits 30:0 of IOADDR.

3. IOADDR is cleared (all bits [31:0]) 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 IOADDR, where:
   - Bit 31 (IOADDR.*Configuration IO Access Enable*) of IOADDR should be set to 1b.
   - Bits 30:0 of IOADDR should hold the actual address of the internal register being read.
2. CSR value is read from IODATA.
   - IODATA is used as a window to the register address specified by IOADDR. As a result, the data read from IODATA is the data of the CSR pointed to by Bits 30:0 of IOADDR.
3. IOADDR is cleared (all bits [31:0]) to avoid unintentional CSR read operations (that might cause clear by read) by other applications scanning the configuration space.

Notes:

- In the event that the GLPCI_CAPSUP.*CSR_CONF_EN* bit is cleared, accesses to IOADDR and IODATA via the configuration address space are ignored and have no effect on the register and the CSRs referenced by IOADDR.
- When functioning in a D3 state, software should not attempt to access CSRs via IOADDR and IODATA.
- To enable CSR access via configuration space, software should set Bit 31 (IOADDR.*Configuration IO Access Enable*) of IOADDR to 1b. Software should clear Bit 31 of IOADDR after completing CSR access to avoid an unintentional clear-by-read operation by another application scanning the configuration address space. Software should also clear Bits 30:0 of IOADDR to remove any trace of previous accesses to the configuration space (see previous flows).
- Bit 31 of IOADDR (IOADDR.*Configuration IO Access Enable*) has no effect when initiating access via I/O address space.

13.1.2 Memory BAR

13.1.2.1 PF BAR Structure

The memory BAR provides access to internal CSRs, to the protocol engine doorbells, to page separated doorbells, and to the external Flash (NVM). This section describes where each of these is located within the BAR space.

The following configuration parameters define the structure of the PF memory BAR 0:

- *FLASH_EXPOSE* bit from the NVM (or the GLPCI_LBARCTRL.*FLASH_EXPOSE* CSR bit).

  0b = Flash memory is not mapped in the memory BAR.

  1b = Flash memory is mapped in the memory BAR. Hardware default; Flash memory is exposed when during initialization the Flash is found to be blank or in error. The Flash area is 16 MB.

- *PE_DB_SIZE* field from the NVM (or the GLPCI_LBARCTRL.*PE_DB_SIZE* CSR field) — Determines the size of the memory space allocated to protocol engine doorbells.

  00b = Memory space is not allocated for PE doorbells.

  01b = A 64 KB area is allocated.

  10b = A (4 MB + 64 KB) area is allocated.

  11b = Reserved.

- GLPCI_LBARCTL.*PAGES_SPACE_EN_PF* enable bit loaded from NVM:

  0b = No region is allocated for SR-IOV mode specific addresses.

  1b = A 96 MB region is allocated.

Table 13-4 lists all supported partitions of the memory BAR as a function of the previously-described parameters. Other combinations of the parameters (not covered in the table) are not supported and considered reserved for future expansion. The following rules apply:

- CSR space is located from the beginning of the BAR until address (8 MB - 64 KB-1).
- PE space (if it exists) is located from address (8 MB - 64 KB) and extends up to address 8 MB, or up to address 16 MB (not including).
- The Flash space (if exposed) is always 16 MB and is in the 16 MB - 32 MB area.
- In scalable I/O mode, the range of 32 MB to 128 MB is used to expose resources in individual 4K pages for user mode access.

The default configuration of the memory BAR (like when the Flash is empty) is defined by hardware default values of the read-only GLPCI_LBARCTRL CSR. GLPCI_LBARCTRL is loaded from the NVM during normal operation (such as when Flash contents are valid).

Table 13-4. Structure of the PF Memory BAR

Flash Exposed	PE_DB Size	SIOV	PE Space		Flash Space		SIOV Space		BAR Size
			Min Addr	Max Addr	Min Addr	Max Addr	Min Addr	Max Addr	
No	00	0 (no SIOV mode)	x	x	x	x	x	x	8 MB
	01		8 MB-64 KB	8 MB					8 MB
	10		8 MB-64 KB	16 MB					16 MB
Yes	00		x	x	16 MB	32 MB			32 MB
	01		8 MB-64 KB	8 MB	16 MB	32 MB			32 MB
	10		8 MB-64 KB	16 MB	16 MB	32 MB			32 MB
No	00	1 (SIOV mode)	x	x	x	x	32 MB	128 MB	128 MB
	01		8 MB-64 KB	8 MB					128 MB
	10		8 MB-64 KB	16 MB					128 MB
Yes	00		x	x	16 MB	32 MB			128 MB
	01		8 MB-64 KB	8 MB	16 MB	32 MB			128 MB
	10		8 MB-64 KB	16 MB	16 MB	32 MB			128 MB

13.1.2.1.1 PF BAR Extension for Scalable I/O Mode

In order to allow allocation of resources to multiple Assignable Interfaces, these resources are mapped to separate 4K pages.

Table 13-5. Resources per area in PF space

Start Address	End Address	Content	Mapping
32 MByte	35 MByte	768 Mailboxes	4K Page/Queue registers
35 MByte	46 MByte	Spare	
46 MByte	47 MByte	Mapping of MSI-X vectors	Contiguous mapping (12K)
47 MByte	48 MByte	256 tail bump registers of Doorbell queues	4K Page/Queue tail bumps
48 MByte	56 MByte	2048 interrupts (0-2047)	4K Page/Interrupt set
56 MByte	64 MByte	2K tail bump registers of Rx-Queues	4K Page/Queue registers
64 MByte	128 MByte	16K tail bump registers of Tx-Queues	4K Page/Queue registers

13.1.2.2 VF BAR Structure

The VF memory BAR provides access to on-die CSRs and (optionally) to Protocol Engine doorbells for VFs.

The GLPCI_LBARCTRL.*VF_PE_DB_SIZE* CSR field determines whether the regions allocated to the Protocol Engine doorbells are provided with a separate region within the BAR:

00b = Memory space is not allocated for PE doorbells.

01b = An 8 KB area is allocated right after the legacy space (0x10000->0x11FFF).

10b = A 64 KB area is allocated (0x10000->0x20000).

11b = Reserved.

Note: Although only 32 VFs supports RDMA, all the VFs requires the same memory space.

Table 13-6 lists all supported partitions of the memory BAR as function of the previously described parameters.

Table 13-6. Structure of the VF Memory BAR

PAGES_SPACE_EN_VF	*VF_PE_DB Size*	BAR Size
0	00b	Max (64 KB, Page Size)
0	01b/10b	Max (128 KB, Page Size)
1	x	Max (256 KB, Page Size)

13.1.3 The MSI-X BAR

The structure of the MSI-X BAR is described in Section 14.3.3 (for a PF) and Section 14.5.3.1 (for a VF).

13.1.4 CSR Organization and Mapping

This section describes how CSRs are mapped into the PF and VF memory BAR. This section does not apply to the following address space:

- The MSI-X BAR (defined per the PCI specifications).
- The memory space allocated to protocol engine doorbells (see Section 13.1.2.1).

13.1.4.1 Mapping by Scope

Registers are associated with a scope. A scope is a set of attributes for a register that define which functions can access the register and how many instances exist for the register. Table 13-7 lists the different scopes.

Table 13-7. Scope Mapping

Scope	PF/VF	Quantity	Exposure	Comments
GL	PF	1	To all PFs.	
GLVF	PF, VF	1	To all PFs and VFs.	Registers are RO.
PRT	PF	8	Each PF has access to the registers of the port it is associated with.	Registers are shared by all PFs on a port.
PRTVF	PF, VF	8	Each PF or VF has access to the registers of the port it is associated with (for a VF, the port is the port the PF is associated with).	Registers are RO. Registers are shared by all PFs and VFs on a port.
PF	PF	8	Each PF has access to its copy only.	
VF[1]	PF, VF	256	Each PF has access to the registers of its VFs only. Each VF has access to its copy only.	
VF16	PF, VF	16	Each PF has access to the registers of its VFs only. Each VF has access to its copy only. Available only for the first 16 VFs.	
VF128	PF, VF	16	Each PF has access to the registers of its VFs only. Each VF has access to its copy only. Available only for the first 128 VFs.	
VP	PF	256	Each PF has access to the registers of its VFs only.	These registers control VF functionality.
VP16	PF	16	Each PF has access to the registers of its VFs only. Available only for the first 16 VFs.	These registers control VF functionality.
VP128	PF	128	Each PF has access to the registers of its VFs only. Available only for the first 128 VFs.	These registers control VF functionality.
QRX	PF, VF	2048	Each PF has access to the registers allocated to it (including its VFs). Each VF has access to the registers allocated to it.	
QTX	PF, VF	16384	Each PF has access to the registers allocated to it (including its VFs). Each VF has access to the registers allocated to it.	
DBLQ	PF, VF	256	Each PF has access to the registers allocated to it (including its VFs). Each VF has access to the registers allocated to it.	
CQ	PF, VF	512	Each PF has access to the registers allocated to it (including its VFs). Each VF has access to the registers allocated to it.	

Table 13-7. Scope Mapping [continued]

Scope	PF/VF	Quantity	Exposure	Comments
VSI	PF	768	All PFs.	Registers are shared by all PFs.
INT	PF	2048	Each PF has access to the registers allocated to it.	
INTVF	PF, VF	512	Each VF has access to the registers allocated to it.[1] Each PF has access to the registers of its VFs only.[1]	

1. PE VF registers are a special case of VF registers. PE resources are provided only to a subset of the VFs (total of 32 VFs). However, all VFs have PE registers mapped in their address space. When a VF accesses a PE register, it is mapped into the internal address space as described here, followed by another translation into one of the 32 physical PE instances. This later translation is not described in this section.

13.1.5 Register Conventions

All registers in the E810 are defined to be 32 bits, and should be accessed as 32-bit DWords. There are some exceptions to this rule:

- Register pairs where two 32-bit registers make up a larger 64-bit logical unit.
- Accesses to Flash memory (via Expansion ROM space, secondary BAR space, or the I/O space) might be byte, word or double word accesses. I/O accesses are limited to DWord accesses (see Section 13.1.1.5).
- Accesses to BAR0 of a VF might be byte, word or DWord accesses. 64-bit (QWord) accesses to this BAR are completed with a Completer Abort (CA) error.
- Access to the MSI-X BAR of the PFs and the VFs might be DWord or QWord accesses.

Reserved bit positions:

Some registers contain certain bits that are marked as reserved. Writes to a reserved field must set the field to its initial value unless specified differently in the field description. Reads from registers containing reserved bits might return indeterminate values in the reserved bit positions unless read values are explicitly stated. When read, these reserved bits should be ignored by software.

Reserved and/or undefined addresses:

Any register address not explicitly declared in this Datasheet are considered to be reserved, and should not be written to. Writing to reserved or undefined register addresses might cause indeterminate behavior. Reads from reserved or undefined configuration register addresses might return indeterminate values unless read values are explicitly stated for specific addresses.

Initial values:

Most registers define the initial hardware values prior to being programmed. In some cases, hardware initial values are undefined and is listed as such via the text undefined, unknown, or X. Such configuration values might need to be set via NVM configuration or via software for proper operation to occur; this need is dependent on the function of the bit. Other registers might cite a hardware default that is overridden by a higher-precedence operation. Operations that might supersede hardware defaults might include a valid NVM load, completion of a hardware operation (such as hardware auto-negotiation), or writing of a different register whose value is then reflected in another bit.

For registers that should be accessed as 32-bit DWords, partial writes (less than a 32-bit DWord) does not take effect (the write is ignored). Partial reads returns all 32 bits of data regardless of the byte enables.

Notes:

- Partial reads to clear-on-read registers (ICR) can have unexpected results since all 32 bits are actually read regardless of the byte enables. Partial reads should not be done.
- All statistics registers are implemented as 32-bit registers. Though some logical statistics registers represent counters in excess of 32 bits in width, registers must be accessed using 32-bit operations (for example, independent access to each 32-bit field). When reading 64-bit statistics registers, the least significant 32-bit register should be read first.

See special notes for VLAN filter table, multicast table arrays, and packet buffer memory that appear in the specific register definitions.

13.1.5.1 Register Abbreviation Naming Conventions

The register abbreviation naming follows (in most cases) the following rules:

- Abbreviation starts with the register's scoping. It could be Port registers (PRT), PF registers (PF) and so on. See Table 13-7 for the complete list of register's scopes.
- Then it follows by the register's main block like DCB (for DCB registers), QF (for queue filters) and so on.
- Then it follows by a few characters that summarize the register's name.

13.1.6 Register Field Attributes

Table 13-8 lists the access type of registers' bit fields. The access rights of the PFs and the VFs to the entire registers are defined per PF and VF registers. In some cases, the PFs and VFs might have Read Only (RO) access rights to registers that can be programmed by the internal logic (either auto-load from the NVM or programmed by the firmware). Registers defined as RO access override any access type defined for its fields.

Table 13-8. Register Field Attributes

Type	Description
RO	**Read Only** A register bit field with this attribute can be read. Writes have no effect on the bit field value. Used also for bits of RO/V type.
RO/V	**Read Only Variant** A register bit field with this attribute can be read. Writes have no effect on the bit field value. The hardware can change the value returned. Software should not expect a static value.
ROCV	**Read Only Clear Value** A register bit field with this attribute can be read. The value returned on the read might be different in each read (typically a counter), and the value is cleared after the read. A write has no effect.
RSV	**Reserved** A register bit field with this attribute can be read and returns an indeterministic value. Writes must set the bit field to its initial value unless specified differently in the field description.
RW	**Read/Write** A register with this attribute can be read and written. Read return the default value, the last value written, or updated status from a previous operation. The field description specifies the field's actual behavior.
RW/V	**Read/Write Variant** Hardware loadable. Software can read from and write to field. Hardware can modify field value. Hardware write have higher priority compare to software write.

Table 13-8. Register Field Attributes [continued]

Type	Description
RCW	**Read Clear/Write** A register bit field with this attribute can be written or read. The value returned on the read might be different than the value written (typically a counter) and the value is cleared after the read.
RW1C	**Read/Write 1 to Clear** A register bit field with this attribute can be read and written. Writing an individual bit within the field to a 1b clears (sets to 0b) the corresponding bit and a write of a 0b has no effect. The value read might return the last value written or the status of a previous operation. The field description specifies the field's actual read behavior.
RW1S	**Read/Write 1 to Set** A register bit field with this attribute can be read and written. Writing an individual bit within the field to a 1b sets (sets to 1b) the corresponding bit and a write of a 0b has no effect. The value read might return the last value written or the status of a previous operation. The field description specifies the field's actual read behavior.
RWC	**Read/Write Clear** A register bit field with this attribute can be written or read. The value returned on the read might be different than the value written (typically a counter), and the value is cleared after a write of any value.
SC	**Self Clear** A register bit field with this attribute can be written or read. he value returned on the read might be different than the value written (typically an action request), as the hardware clears the bit when the action is done/activated.
WO	**Write Only** A register bit field with this attribute can be only written. A read returns a fixed value not reflecting the content of the register. This is used for keys that should be not be visible to other software agents.

13.2 Device Registers - PF

13.2.1 BAR0 Registers Summary

Table 13-9. PF - General Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00074000 + 0x4*VF, VF=0...255	VFGEN_RSTAT[VF]	VF Reset Status	13.2.2.1.1
0x00083048	GL_FWSTS	Firmware Status Register	13.2.2.1.2
0x00088000	PFGEN_STATE	PF State	13.2.2.1.3
0x000880C8 + 0x4*n, n=0...6	GLGEN_GPIO_CTL[n]	Global GPIO Control	13.2.2.1.4
0x00090000 + 0x4*VF, VF=0...255	VPGEN_VFRTRIG[VF]	VF Reset Trigger	13.2.2.1.5
0x00090800 + 0x4*VF, VF=0...255	VPGEN_VFRSTAT[VF]	VF Reset Status	13.2.2.1.6
0x00091000	PFGEN_CTRL	PFGEN Control	13.2.2.1.7
0x00091080	PFGEN_PFRSTAT	PFR STAT	13.2.2.1.8
0x00091180	PFGEN_DRUN	PF Driver Unload	13.2.2.1.9
0x00091800 + 0x4*VSI, VSI=0...767	VSIGEN_RTRIG[VSI]	VM Reset Trigger	13.2.2.1.10
0x00092800 + 0x4*VSI, VSI=0...767	VSIGEN_RSTAT[VSI]	VM Reset Status	13.2.2.1.11
0x00093804	GL_XLR_MARKER_TRIG_VMLR	XLR Marker Trigger	13.2.2.1.12
0x000939E8	GLGEN_MARKER_COUNT	Global Marker Count	13.2.2.1.13
0x000939EC	GLGEN_XLR_TRNS_WAIT_COUNT	Global Wait Between Transaction Count	13.2.2.1.14
0x000939F0	GLGEN_XLR_MSK2HLP_RDY	Global Wait for HLP After CORER	13.2.2.1.15
0x000939F4	GLGEN_ECC_ERR_RST_MASK_L	ECC Error Mask Low	13.2.2.1.16
0x000939F8	GLGEN_ECC_ERR_RST_MASK_H	ECC Error Mask High	13.2.2.1.17
0x000939FC	GLGEN_ECC_ERR_INT_TOG_MASK_L	ECC Error Int Mask Low	13.2.2.1.18
0x00093A00	GLGEN_ECC_ERR_INT_TOG_MASK_H	ECC Error Int Mask High	13.2.2.1.19
0x00093A04 + 0x4*n, n=0...7	GLGEN_VFLRSTAT[n]	PFR STAT	13.2.2.1.20
0x000A2000	GL_XLR_MARKER_TRIG_TCVMLR	TCVMLR XLR Marker Trigger	13.2.2.1.21
0x000A2004	GL_TCVMLR_QCTL	Transmit Scheduler Queue Control	13.2.2.1.22
0x000A2008	GL_TCVMLR_DRAIN_MARKER	TCVMLR Drain Marker Control	13.2.2.1.23
0x000A200C	GL_TCVMLR_QCNTR	TCVMLR Halt Done Down Counter	13.2.2.1.24
0x000A2010	GL_TCVMLR_QCFG	TCVMLR Queue Port TC Config Control	13.2.2.1.25
0x000A2014	GL_TCVMLR_QCFG_RD	TCVMLR Queue Port TC Config Status	13.2.2.1.26
0x000A2018	GL_TCVMLR_REQ_STAT	TCVMLR Req Flow Status Control	13.2.2.1.27
0x000A201C	GL_TCVMLR_STAT	TCVMLR Req Flow Status Read	13.2.2.1.28
0x000A2024	GL_TCVMLR_ERR_STAT	TCVMLR Req Flow Error Status	13.2.2.1.29

Table 13-9. PF - General Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x000A20A8 + 0x4*n, n=0...31	GL_TCVMLR_DRAIN_DONE_TCLAN[n]	TCVMLR Drain Done Count for TCLAN	13.2.2.1.30
0x000A2128 + 0x4*n, n=0...31	GL_TCVMLR_DRAIN_DONE_TPB[n]	TCVMLR Drain Done Count for TPB	13.2.2.1.31
0x000A21A8	GL_TCVMLR_DRAIN_DONE_DEC	TCVMLR Drain Done Decrement Control	13.2.2.1.32
0x000A21C0	PRT_TCVMLR_DRAIN_CNTR	TCVMLR Drain Done Down Counter	13.2.2.1.33
0x000A21E0	GL_TCVMLR_DRAIN_CNTR_CTL	TCVMLR Drain Done Down Counter Control	13.2.2.1.34
0x000B612C	GLGEN_STAT	Global Status	13.2.2.1.35
0x000B8100	PRTGEN_STATUS	General Port Status	13.2.2.1.36
0x000B8120	PRTGEN_CNF	General Port Configuration	13.2.2.1.37
0x000B8160	PRTGEN_CNF2	General Port Configuration2	13.2.2.1.38
0x000B8180	GLGEN_RSTCTL	Global Reset Control	13.2.2.1.39
0x000B8184	GLGEN_CLKSTAT	Global Clock Status	13.2.2.1.40
0x000B8188	GLGEN_RSTAT	Global Reset Status	13.2.2.1.41
0x000B8190	GLGEN_RTRIG	Global Reset Trigger	13.2.2.1.42
0x000B81E4	GLGEN_ASSERT_HLP	Global Switch Mode Reset Control	13.2.2.1.43
0x000B8214	GLVFGEN_TIMER	Global Device Timer	13.2.2.1.44
0x000B826C	GLGEN_CLKSTAT_SRC	Global Clock Status	13.2.2.1.45
0x000B8280	PRTGEN_CNF3	General Port Configuration3	13.2.2.1.46
0x001D2400	PFGEN_PORTNUM	LAN Port Number	13.2.2.1.47
0x0020C000 + 0x4*n, n=0...63	GLGEN_ANA_FLAG_MAP[n]	GLGEN_ANA_FLAG_MAP	13.2.2.1.48
0x0020C100	GLGEN_ANA_DEF_PTYPE	GLGEN_ANA_DEF_PTYPE	13.2.2.1.49
0x0020C104	GLGEN_ANA_CFG_CTRL	GLGEN_ANA_CFG_CTRL	13.2.2.1.50
0x0020C108	GLGEN_ANA_CFG_WRDATA	GLGEN_ANA_CFG_WRDATA	13.2.2.1.51
0x0020C10C + 0x4*n, n=0...15	GLGEN_ANA_CFG_RDDATA[n]	GLGEN_ANA_CFG_RDDATA	13.2.2.1.52
0x0020C14C + 0x4*n, n=0...2	GLGEN_ANA_CFG_LU_KEY[n]	GLGEN_ANA_CFG_LU_KEY	13.2.2.1.53
0x0020C158	GLGEN_ANA_CFG_HTBL_LU_RESULT	GLGEN_ANA_CFG_HTBL_LU_RESULT	13.2.2.1.54
0x0020C15C	GLGEN_ANA_CFG_SPLBUF_LU_RESULT	GLGEN_ANA_CFG_SPLBUF_LU_RESULT	13.2.2.1.55
0x0020C160 + 0x4*n, n=0...15	GLGEN_ANA_P2P[n]	GLGEN_ANA_P2P	13.2.2.1.56
0x0020C1A0 + 0x4*n, n=0...3	GLGEN_ANA_PG0_HASHKEY[n]	GLGEN_ANA_PG0_HASHKEY	13.2.2.1.57
0x0020C1B0 + 0x4*n, n=0...3	GLGEN_ANA_NMPG0_HASHKEY[n]	GLGEN_ANA_NMPG0_HASHKEY	13.2.2.1.58
0x0020C1C0 + 0x4*n, n=0...3	GLGEN_ANA_PG_KEYMASK[n]	GLGEN_ANA_PG_KEYMASK	13.2.2.1.59
0x0020C1D0 + 0x4*n, n=0...3	GLGEN_ANA_NMPG_KEYMASK[n]	GLGEN_ANA_NMPG_KEYMASK	13.2.2.1.60

Table 13-9. PF - General Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0020C1E4 + 0x4*n, n=0...5	GLGEN_ANA_LAST_PROT_ID[n]	GLGEN_ANA_LAST_PROT_ID	13.2.2.1.61
0x0020C1FC	GLGEN_ANA_PROFIL_CTRL	GLGEN_ANA_PROFIL	13.2.2.1.62
0x0020C200	GLGEN_ANA_OUT_OF_PKT	GLGEN_ANA_OUT_OF_PKT	13.2.2.1.63
0x0020C204	GLGEN_ANA_NO_HIT_PG_NM_PG	GLGEN_ANA_NO_HIT_PG_NM_PG	13.2.2.1.64
0x0020C208	GLGEN_ANA_ALU_ACCSS_OUT_OF_PKT	GLGEN_ANA_ALU_ACCSS_OUT_OF_PKT	13.2.2.1.65
0x0020C210	GLGEN_ANA_INV_NODE_PTYPE	GLGEN_ANA_INV_NODE_PTYPE	13.2.2.1.66
0x0020C218	GLGEN_ANA_INV_PTYPE_MARKER	GLGEN_ANA_INV_PTYPE_MARKER	13.2.2.1.67
0x0020C21C	GLGEN_ANA_ABORT_PTYPE	GLGEN_ANA_ABORT_PTYPE	13.2.2.1.68
0x0020C220	GLGEN_ANA_ERR_CTRL	GLGEN_ANA_ERR_CTRL	13.2.2.1.69
0x0020D000 + 0x4*n, n=0...63	GLGEN_ANA_TX_FLAG_MAP[n]	GLGEN_ANA_TX_FLAG_MAP	13.2.2.1.70
0x0020D100	GLGEN_ANA_TX_DEF_PTYPE	GLGEN_ANA_TX_DEF_PTYPE	13.2.2.1.71
0x0020D104	GLGEN_ANA_TX_CFG_CTRL	GLGEN_ANA_TX_CFG_CTRL	13.2.2.1.72
0x0020D108	GLGEN_ANA_TX_CFG_WRDATA	GLGEN_ANA_TX_CFG_WRDATA	13.2.2.1.73
0x0020D10C + 0x4*n, n=0...15	GLGEN_ANA_TX_CFG_RDDATA[n]	GLGEN_ANA_TX_CFG_RDDATA	13.2.2.1.74
0x0020D14C + 0x4*n, n=0...2	GLGEN_ANA_TX_CFG_LU_KEY[n]	GLGEN_ANA_TX_CFG_LU_KEY	13.2.2.1.75
0x0020D158	GLGEN_ANA_TX_CFG_HTBL_LU_RESULT	GLGEN_ANA_TX_CFG_HTBL_LU_RESULT	13.2.2.1.76
0x0020D15C	GLGEN_ANA_TX_CFG_SPLBUF_LU_RESULT	GLGEN_ANA_TX_CFG_SPLBUF_LU_RESULT	13.2.2.1.77
0x0020D160 + 0x4*n, n=0...15	GLGEN_ANA_TX_P2P[n]	GLGEN_ANA_TX_P2P	13.2.2.1.78
0x0020D1A0 + 0x4*n, n=0...3	GLGEN_ANA_TX_PG0_HASHKEY[n]	GLGEN_ANA_TX_PG0_HASHKEY	13.2.2.1.79
0x0020D1B0 + 0x4*n, n=0...3	GLGEN_ANA_TX_NMPG0_HASHKEY[n]	GLGEN_ANA_TX_NMPG0_HASHKEY	13.2.2.1.80
0x0020D1C0 + 0x4*n, n=0...3	GLGEN_ANA_TX_PG_KEYMASK[n]	GLGEN_ANA_TX_PG_KEYMASK	13.2.2.1.81
0x0020D1D0 + 0x4*n, n=0...3	GLGEN_ANA_TX_NMPG_KEYMASK[n]	GLGEN_ANA_TX_NMPG_KEYMASK	13.2.2.1.82
0x0020D1FC	GLGEN_ANA_TX_PROFIL_CTRL	GLGEN_ANA_TX_PROFIL_CTRL	13.2.2.1.83
0x0020D204	GLGEN_ANA_TX_NO_HIT_PG_NM_PG	GLGEN_ANA_TX_NO_HIT_PG_NM_PG	13.2.2.1.84
0x0020D208	GLGEN_ANA_TX_ALU_ACCSS_OUT_OF_PKT	GLGEN_ANA_TX_ALU_ACCSS_OUT_OF_PKT	13.2.2.1.85
0x0020D210	GLGEN_ANA_TX_INV_NODE_PTYPE	GLGEN_ANA_TX_INV_NODE_PTYPE	13.2.2.1.86
0x0020D214	GLGEN_ANA_TX_INV_PROT_ID	GLGEN_ANA_TX_INV_PROT_ID	13.2.2.1.87
0x0020D218	GLGEN_ANA_TX_INV_PTYPE_MARKER	GLGEN_ANA_TX_INV_PTYPE_MARKER	13.2.2.1.88
0x0020D21C	GLGEN_ANA_TX_ABORT_PTYPE	GLGEN_ANA_TX_ABORT_PTYPE	13.2.2.1.89
0x0020D220	GLGEN_ANA_TX_ERR_CTRL	GLGEN_ANA_TX_ERR_CTRL	13.2.2.1.90
0x0020D4CC	GLGEN_ANA_TX_DFD_PACE_OUT	GLGEN_ANA_TX_DFD_PACE_OUT	13.2.2.1.91

Table 13-10. PF - Internal Fuses Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x000A400C	GL_UFUSE_SOC	SKU Fuses	13.2.2.2.1

Table 13-11. PF - PCIe Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0009D880	PFPCI_SUBSYSID	PFPCIe Subsystem ID	13.2.2.3.1
0x0009D980	PFPCI_FUNC	PCIe Functions Configuration	13.2.2.3.2
0x0009DA00	PFPCI_STATUS1	PCIe Function Status 1	13.2.2.3.3
0x0009DA80	PFPCI_PM	PCIe PM	13.2.2.3.4
0x0009DB00	PFPCI_CLASS	PCIe Storage Class	13.2.2.3.5
0x0009DE00	PFPCI_DEVID	PCIe PF Device ID	13.2.2.3.6
0x0009DE70	GL_CLKGATE_EVENTS	Clock Gating Events	13.2.2.3.7
0x0009DE74	GLPCI_LBARCTRL	PCI BAR Control	13.2.2.3.8
0x0009DE7C	GLPCI_PWRDATA	PCIe Power Data Register	13.2.2.3.9
0x0009DE80	GLPCI_SERL	PCIe Serial Number MAC Address Low	13.2.2.3.10
0x0009DE84	GLPCI_SERH	PCIe Serial Number MAC Address High	13.2.2.3.11
0x0009DE88	GLPCI_CAPCTRL	PCIe Capabilities Control	13.2.2.3.12
0x0009DE8C	GLPCI_CAPSUP	PCIe Capabilities Support	13.2.2.3.13
0x0009DE90	GLPCI_LINKCAP	PCIe Link Capabilities	13.2.2.3.14
0x0009DE94	GLPCI_PMSUP	PCIe PM Support	13.2.2.3.15
0x0009DE98	GLPCI_REVID	PCIe Revision ID	13.2.2.3.16
0x0009DE9C	GLPCI_VFSUP	PCIe VF Capabilities Support	13.2.2.3.17
0x0009DEA0	GLPCI_CNF	PCIe Global Config	13.2.2.3.18
0x0009DEC8	GLPCI_VENDORID	PCIe Vendor ID	13.2.2.3.19
0x0009DEE8	GLPCI_SUBVENID	PCIe Subsystem ID	13.2.2.3.20
0x0009DF00	PFPCI_CNF	PCIe PF Configuration	13.2.2.3.21
0x0009DF40	PQ_FIFO_STATUS	Posted Queue IOSF FIFO Status	13.2.2.3.22
0x0009DF44	GLPCI_PUSH_PE_IF_TO_STATUS	Push PE IF Status	13.2.2.3.23
0x0009E000 + 0x4*VF, VF=0...255	PFPCI_VF_FLUSH_DONE[VF]	PCIe VF Flush Done	13.2.2.3.24
0x0009E400	PFPCI_PF_FLUSH_DONE	PCIe PF Flush Done	13.2.2.3.25
0x0009E480	PFPCI_VM_FLUSH_DONE	PCIe VM Flush Done	13.2.2.3.26
0x0009E500	PF_PCI_CIAD	PCIe Configuration Indirect Access Data	13.2.2.3.27
0x0009E580	PF_PCI_CIAA	PCIe Configuration Indirect Access Address	13.2.2.3.28
0x0009E600	PFPCI_VMINDEX	PCIe VM Pending Index	13.2.2.3.29
0x0009E800	PFPCI_VMPEND	PCIe VM Pending Status	13.2.2.3.30
0x0009E880	PF_FUNC_RID	Function Requester ID Information Register	13.2.2.3.31
0x0009E900	PFPCI_FACTPS	Function Active and Power State	13.2.2.3.32

Table 13-11. PF - PCIe Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0009E954 + 0x4*n, n=0...3	GLPCI_GSCL_5_8[n]	PCIe Statistic Control Register #5...#8	13.2.2.3.33
0x0009E970	GLPCI_BYTCTH_P	PCIe Byte Counter High	13.2.2.3.34
0x0009E994	GLPCI_BYTCTL_P	PCIe Byte Counter Low	13.2.2.3.35
0x0009E998	GLPCI_GSCL_2	PCIe Statistic Control Registers #2	13.2.2.3.36
0x0009E99C + 0x4*n, n=0...3	GLPCI_GSCN_0_3[n]	PCIe Statistic Counter Registers #0...#3	13.2.2.3.37
0x0009E9AC	GLPCI_DREVID	PCIe Default Revision ID	13.2.2.3.38
0x0009E9B0	GLPCI_PKTCT_P	PCIe Packet Counter	13.2.2.3.39
0x0009E9B4	GLPCI_GSCL_1_P	PCIe Statistic Control Register #1	13.2.2.3.40
0x000BE004	GLPCI_CNF2	PCIe Global Config 2	13.2.2.3.41
0x000BE0D4	GLPCI_UPADD	PCIe Upper Address	13.2.2.3.42
0x000BFD80	GLPCI_NPQ_CFG	PCIe NPQ Config	13.2.2.3.43
0x000BFD90	GLPCI_WATMK_CLNT_PIPEMON	PCIe NPQ Watermark of Pipe Monitor	13.2.2.3.44
0x000BFD9C	GLPCI_PKTCT_NP_C	PCIe Packet Counter	13.2.2.3.45
0x000BFDA0	GLPCI_LATCT_NP_C	PCIe Packet Counter	13.2.2.3.46
0x000BFDA4	GLPCI_GSCL_1_NP_C	PCIe Statistic Control Register #1	13.2.2.3.47
0x000BFDA8	GLPCI_BYTCTH_NP_C	PCIe Byte Counter High	13.2.2.3.48
0x000BFDAC	GLPCI_BYTCTL_NP_C	PCIe Byte Counter Low	13.2.2.3.49

Table 13-12. PF - MAC Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x001E3180	PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE	HSEC CONTROL Receive PFC ENABLE	13.2.2.4.1
0x001E31A0	PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE	HSEC CONTROL Transmit PAUSE_ENABLE	13.2.2.4.2
0x001E31C0	PRTMAC_HSEC_CTL_RX_ENABLE_GCP	HSEC CONTROL Receive ENABLE_GCP	13.2.2.4.3
0x001E3220	PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART1	HSEC CONTROL Receive PAUSE_DA_UCAST_PART1	13.2.2.4.4
0x001E3240	PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART2	HSEC CONTROL Receive PAUSE_DA_UCAST_PART2	13.2.2.4.5
0x001E3280	PRTMAC_HSEC_CTL_RX_PAUSE_SA_PART1	HSEC CONTROL Receive PAUSE_SA_PART1	13.2.2.4.6
0x001E32A0	PRTMAC_HSEC_CTL_RX_PAUSE_SA_PART2	HSEC CONTROL Receive PAUSE_SA_PART2	13.2.2.4.7
0x001E34C0	PRTMAC_HSEC_CTL_RX_ENABLE_GPP	HSEC CONTROL Receive ENABLE_GPP	13.2.2.4.8
0x001E35C0	PRTMAC_HSEC_CTL_RX_ENABLE_PPP	HSEC CONTROL Receive ENABLE_PPP	13.2.2.4.9
0x001E36C0	PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL	HSEC CONTROL Receive FORWARD_CONTROL	13.2.2.4.10
0x001E36E0 + 0x20*n, n=0...8	PRTMAC_HSEC_CTL_TX_PAUSE_QUANTA[n]	HSEC CONTROL Transmit PAUSE_QUANTA	13.2.2.4.11

Table 13-12. PF - MAC Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x001E3800 + 0x20*n, n=0...8	PRTMAC_HSEC_CTL_TX_PAUSE_REFRESH_TIMER[n]	HSEC CONTROL Transmit PAUSE_REFRESH_TIMER	13.2.2.4.12
0x001E3960	PRTMAC_HSEC_CTL_TX_SA_PART1	HSEC CONTROL Transmit SA_GPP_PART1	13.2.2.4.13
0x001E3980	PRTMAC_HSEC_CTL_TX_SA_PART2	HSEC CONTROL Transmit SA_GPP_PART2	13.2.2.4.14
0x001E3C20	PRTMAC_RX_PKT_DRP_CNT	MAC Rx Silent Drop Count	13.2.2.4.15
0x001E3C40	PRTMAC_HSEC_CTL_RX_QUANTA_SHIFT	MAC Rx Shift FC Quanta	13.2.2.4.16
0x001E3C60 + 0x20*n, n=0...7	PRTMAC_MD_OVRRIDE_ENABLE[n]	MAC Rx Metadata Override Enable	13.2.2.4.17
0x001E3D60 + 0x20*n, n=0...7	PRTMAC_MD_OVRRIDE_VAL[n]	MAC Rx Metadata Override Value	13.2.2.4.18
0x001E47C0	PRTMAC_LINK_DOWN_COUNTER	Link Down Counter	13.2.2.4.19
0x001E4840	PRTMAC_TX_LNK_UP_CNT	Link UP Counter Limit	13.2.2.4.20
0x001E48C0	PRTMAC_TX_CNT_MRKR	MAC Markers Counter Tx	13.2.2.4.21
0x001E48E0	PRTMAC_RX_CNT_MRKR	MAC Markers Counter Rx	13.2.2.4.22

Table 13-13. PF - Power Management Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x000B81BC	GLGEN_PME_TO	PME_TO Indication	13.2.2.5.1
0x000B81EC	GL_PWR_MODE_DIVIDE_S5_H_CTRL	Global Power Mode Control S5	13.2.2.5.2
0x000B81F0	GL_PWR_MODE_DIVIDE_S0_CTRL_H_PECLK	Global Power Mode Control PE	13.2.2.5.3
0x000B81F4	GL_PWR_MODE_DIVIDE_S0_CTRL_H_UCLK	Global Power Mode Control Upper	13.2.2.5.4
0x000B81F8	GL_PWR_MODE_DIVIDE_S0_CTRL_H_RXCTL	Global Power Mode Control RXCTL	13.2.2.5.5
0x000B81FC	GL_PWR_MODE_DIVIDE_S0_CTRL_H_PSM	Global Power Mode Control PSM	13.2.2.5.6
0x000B8200	GL_PWR_MODE_DIVIDE_S0_CTRL_H_LCLK	Global Power Mode Control Lower	13.2.2.5.7
0x000B8208	GL_PWR_MODE_DIVIDE_S0_CTRL_H_UANA	Global Power Mode Control UANA	13.2.2.5.8
0x000B820C	GL_PWR_MODE_CTL	Global Power Mode Control	13.2.2.5.9
0x000B8218	GL_PWR_MODE_DIVIDE_CTRL_L_DEFAULT	Global Power Mode Control Defaults	13.2.2.5.10
0x000B821C	GL_PWR_MODE_DIVIDE_S0_CTRL_M_PECLK	Global Power Mode Control PE	13.2.2.5.11
0x000B8220	GL_PWR_MODE_DIVIDE_S0_CTRL_L_PECLK	Global Power Mode Control PE	13.2.2.5.12
0x000B8224	GL_PWR_MODE_DIVIDE_S0_CTRL_M_UCLK	Global Power Mode Control Upper	13.2.2.5.13
0x000B8228	GL_PWR_MODE_DIVIDE_S0_CTRL_M_RXCTL	Global Power Mode Control RXCTL	13.2.2.5.14
0x000B822C	GL_PWR_MODE_DIVIDE_S0_CTRL_M_PSM	Global Power Mode Control PSM	13.2.2.5.15
0x000B8230	GL_PWR_MODE_DIVIDE_S0_CTRL_M_LCLK	Global Power Mode Control Lower	13.2.2.5.16
0x000B8234	GL_PWR_MODE_DIVIDE_S0_CTRL_M_UANA	Global Power Mode Control UANA	13.2.2.5.17
0x000B8238	GL_PWR_MODE_DIVIDE_S0_CTRL_L_UCLK	Global Power Mode Control Upper	13.2.2.5.18
0x000B823C	GL_PWR_MODE_DIVIDE_S0_CTRL_L_RXCTL	Global Power Mode Control RXCTL	13.2.2.5.19
0x000B8240	GL_PWR_MODE_DIVIDE_S0_CTRL_L_PSM	Global Power Mode Control PSM	13.2.2.5.20

Table 13-13. PF - Power Management Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x000B8244	GL_PWR_MODE_DIVIDE_S0_CTRL_L_LCLK	Global Power Mode Control Lower	13.2.2.5.21
0x000B8248	GL_PWR_MODE_DIVIDE_S0_CTRL_L_UANA	Global Power Mode Control UANA	13.2.2.5.22
0x000B824C	GL_PWR_MODE_DIVIDE_S5_L_CTRL	Global Power Mode Control S5	13.2.2.5.23
0x000B8250	GL_PWR_MODE_DIVIDE_S5_M_CTRL	Global Power Mode Control S5	13.2.2.5.24
0x000B825C	GL_PWR_MODE_DIVIDE_CTRL_H_DEFAULT	Global Power Mode Control Defaults	13.2.2.5.25
0x000B8260	GL_PWR_MODE_DIVIDE_CTRL_M_DEFAULT	Global Power Mode Control Defaults	13.2.2.5.26
0x000B8270	GL_S5_PWR_MODE_EXIT_CTL	Global Power Mode Control	13.2.2.5.27
0x001E4320	PRTPM_EEE_STAT	Energy Efficient Ethernet (EEE) Status	13.2.2.5.28
0x001E4360	PRTPM_EEER	Energy Efficient Ethernet (EEE) Register	13.2.2.5.29
0x001E4380	PRTPM_EEEC	Energy Efficient Ethernet (EEE) Control	13.2.2.5.30
0x001E43A0	PRTPM_RLPIC	EEE Rx LPI Count	13.2.2.5.31
0x001E43C0	PRTPM_TLPIC	EEE Tx LPI Count	13.2.2.5.32
0x001E43E0	PRTPM_EEETXC	EEE Tx Control	13.2.2.5.33
0x001E4400	PRTPM_EEEFWD	EEE Tx FW Done	13.2.2.5.34

Table 13-14. PF - Wake-Up Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0009DB80	PFPM_WUS	Wake-Up Status Register	13.2.2.6.1
0x0009DC00	PFPM_WUFC	Wake-Up Filter Control Register	13.2.2.6.2
0x0009DC80	PFPM_WUC	Wake-Up Control Register	13.2.2.6.3
0x0009DEE4	GLPM_WUMC	Wake-Up on MNG Control	13.2.2.6.4
0x000B8080	PFPM_APM	APM Control Register	13.2.2.6.5
0x001E3B20 + 0x20*n, n=0...3	PRTPM_SAL[n]	MAC Address Low	13.2.2.6.6
0x001E3BA0 + 0x20*n, n=0...3	PRTPM_SAH[n]	MAC Address High	13.2.2.6.7

Table 13-15. PF - NVM Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x000824C4	GLNVM_AL_DONE_HLP	HLP Auto-Load Done Register	13.2.2.7.1
0x000B6008	GLNVM_ULD	Unit Load Status	13.2.2.7.2
0x000B6010 + 0x4*n, n=0...59	GLNVM_PROTCSR[n]	Protected CSR List	13.2.2.7.3
0x000B6100	GLNVM_GENS	Global NVM General Status Register	13.2.2.7.4
0x000B6108	GLNVM_FLA	Flash Access Register	13.2.2.7.5
0x000B6140	GLNVM_ALTIMERS	Auto-Load Timers	13.2.2.7.6
0x000B6154	GLNVM_ULT	Unit Load Timeout	13.2.2.7.7

Table 13-16. PF - Analyzer Registers (Pre Parser) Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x000492A8 + 0x4*n, n=0...7	GL_SWT_L2TAG0[n]	L2 Tag Data Low	13.2.2.8.1
0x000492C8 + 0x4*n, n=0...7	GL_SWT_L2TAG1[n]	L2 Tag Data High	13.2.2.8.2
0x000492E8 + 0x4*n, n=0...7	GL_SWT_L2TAGTXIB[n]	L2 Tag Tx Insert Bytes	13.2.2.8.3
0x00052000 + 0x4*n, n=0...7	GL_SWT_L2TAGRXEB[n]	L2 Tag Rx Extract Bytes	13.2.2.8.4
0x001D2660 + 0x4*n, n=0...7	GL_SWT_L2TAGCTRL[n]	L2 Tag Control	13.2.2.8.5

Table 13-17. PF - FlexiPipe Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0020E000 + 0x4*n, n=0...2	GL_PSTEXT_FORCE_PID[n]	Force Profile ID	13.2.2.9.1
0x0020E00C + 0x4*n, n=0...2	GL_PSTEXT_PLVL_SEL[n]	Profile Level Selector	13.2.2.9.2
0x0020E018 + 0x4*n, n=0...2	GL_PSTEXT_FORCE_L1CDID[n]	L1 Force CDID	13.2.2.9.3
0x0020E024 + 0x4*n, n=0...2	GL_PSTEXT_P2P_L1ADDR[n]	L1 P2P Table Configuration Address	13.2.2.9.4
0x0020E030 + 0x4*n, n=0...2	GL_PSTEXT_P2P_L1DATA[n]	L1 P2P Table Configuration Data	13.2.2.9.5
0x0020E03C + 0x4*n, n=0...2	GL_PSTEXT_XLT0_L1ADDR[n]	XLT0 Table Configuration Address	13.2.2.9.6
0x0020E048 + 0x4*n, n=0...2	GL_PSTEXT_XLT0_L1DATA[n]	XLT0 Table Configuration Data	13.2.2.9.7
0x0020E054 + 0x4*n, n=0...2	GL_PSTEXT_CDMD_L1SEL[n]	L1 Bidirectional CTL	13.2.2.9.8
0x0020E060 + 0x4*n, n=0...2	GL_PSTEXT_FLGS_L1TBL[n]	L1 Flag Select Table	13.2.2.9.9
0x0020E06C + 0x4*n, n=0...2	GL_PSTEXT_FLGS_L1SEL0_1[n]	L1 Flag Select Control (0-1)	13.2.2.9.10
0x0020E078 + 0x4*n, n=0...2	GL_PSTEXT_FLGS_L1SEL2_3[n]	L1 Flag Select Control (2-3)	13.2.2.9.11
0x0020E084 + 0x4*n, n=0...2	GL_PSTEXT_CTLTBL_L2ADDR[n]	L2 Configuration Table Address	13.2.2.9.12
0x0020E090 + 0x4*n, n=0...2	GL_PSTEXT_CTLTBL_L2DATA[n]	L2 Configuration Table Data	13.2.2.9.13
0x0020E09C + 0x4*n, n=0...2	GL_PSTEXT_L2PRTMOD[n]	XLT1, XLT2 Partition Mode	13.2.2.9.14
0x0020E0C0 + 0x4*n, n=0...2	GL_PSTEXT_XLT1_L2ADDR[n]	XLT1 Table Configuration Address	13.2.2.9.15
0x0020E0CC + 0x4*n, n=0...2	GL_PSTEXT_XLT1_L2DATA[n]	XLT1 Table Configuration Data	13.2.2.9.16
0x0020E0D8 + 0x4*n, n=0...2	GL_PSTEXT_XLT2_L2ADDR[n]	XLT2 Table Configuration Address	13.2.2.9.17
0x0020E0E4 + 0x4*n, n=0...2	GL_PSTEXT_XLT2_L2DATA[n]	XLT2 Table Configuration Data	13.2.2.9.18
0x0020E0F0 + 0x4*n, n=0...2	GL_PSTEXT_PID_L2GKTYPE[n]	Profile ID Gen Key Type	13.2.2.9.19
0x0020E0FC + 0x4*n, n=0...2	GL_PSTEXT_L2_PMASK0[n]	Profile Key Mask (LSB)	13.2.2.9.20
0x0020E108 + 0x4*n, n=0...2	GL_PSTEXT_L2_PMASK1[n]	Profile Key Mask (MSB)	13.2.2.9.21
0x0020E114 + 0x4*n, n=0...2	GL_PSTEXT_TCAM_L2ADDR[n]	TCAM Configuration (Address)	13.2.2.9.22
0x0020E120 + 0x4*n, n=0...2	GL_PSTEXT_TCAM_L2DATALSB[n]	TCAM Configuration LSB (Data)	13.2.2.9.23
0x0020E12C + 0x4*n, n=0...2	GL_PSTEXT_TCAM_L2DATAMSB[n]	TCAM Configuration MSB (Data+Mask)	13.2.2.9.24
0x0020E138 + 0x4*n, n=0...2	GL_PSTEXT_DFLT_L2PRFL[n]	L2 Default Profile	13.2.2.9.25
0x0020E144 + 0x4*n, n=0...2	GL_PSTEXT_K2N_L2ADDR[n]	K2N Table Configuration Address	13.2.2.9.26
0x0020E150 + 0x4*n, n=0...2	GL_PSTEXT_K2N_L2DATA[n]	K2N Table Configuration Data	13.2.2.9.27
0x0020E15C + 0x4*n, n=0...2	GL_PSTEXT_N2N_L2ADDR[n]	K2N Table Configuration Address	13.2.2.9.28

Table 13-17. PF - FlexiPipe Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0020E168 + 0x4*n, n=0...2	GL_PSTEXT_N2N_L2DATA[n]	K2N Table Configuration Data	13.2.2.9.29
0x0020E174 + 0x4*n, n=0...63	GL_PSTEXT_PRFLM_DATA_0[n]	Profile Memory Configuration Data	13.2.2.9.30
0x0020E274 + 0x4*n, n=0...63	GL_PSTEXT_PRFLM_DATA_1[n]	Profile Memory Configuration Data	13.2.2.9.31
0x0020E374 + 0x4*n, n=0...63	GL_PSTEXT_PRFLM_DATA_2[n]	Profile Memory Configuration Data	13.2.2.9.32
0x0020E474 + 0x4*n, n=0...2	GL_PSTEXT_PRFLM_CTRL[n]	Profile Memory Configuration Control	13.2.2.9.33
0x0020E480 + 0x4*n, n=0...2	GL_PSTEXT_FL15_BMPLSB[n]	PG L2 Flag15 Bitmask (LSB)	13.2.2.9.34
0x0020E48C + 0x4*n, n=0...2	GL_PSTEXT_FL15_BMPMSB[n]	PG L2 Flag15 Bitmask (MSB)	13.2.2.9.35
0x0020E498 + 0x4*n, n=0...2	GL_PSTEXT_L2_TMASK0[n]	Profile Key Mask (LSB)	13.2.2.9.36
0x0020E4A4 + 0x4*n, n=0...2	GL_PSTEXT_L2_TMASK1[n]	Profile Key Mask (MSB)	13.2.2.9.37
0x0020F000 + 0x4*n, n=0...2	GL_PREEXT_FORCE_PID[n]	Force Profile ID	13.2.2.9.38
0x0020F00C + 0x4*n, n=0...2	GL_PREEXT_PLVL_SEL[n]	Profile Level Selector	13.2.2.9.39
0x0020F018 + 0x4*n, n=0...2	GL_PREEXT_FORCE_L1CDID[n]	L1 Force CDID	13.2.2.9.40
0x0020F024 + 0x4*n, n=0...2	GL_PREEXT_P2P_L1ADDR[n]	L1 P2P Table Configuration Address	13.2.2.9.41
0x0020F030 + 0x4*n, n=0...2	GL_PREEXT_P2P_L1DATA[n]	L1 P2P Table Configuration Data	13.2.2.9.42
0x0020F03C + 0x4*n, n=0...2	GL_PREEXT_XLT0_L1ADDR[n]	XLT0 Table Configuration Address	13.2.2.9.43
0x0020F048 + 0x4*n, n=0...2	GL_PREEXT_XLT0_L1DATA[n]	XLT0 Table Configuration Data	13.2.2.9.44
0x0020F054 + 0x4*n, n=0...2	GL_PREEXT_CDMD_L1SEL[n]	L1 Bidirectional CTL	13.2.2.9.45
0x0020F060 + 0x4*n, n=0...2	GL_PREEXT_FLGS_L1TBL[n]	L1 Flag Select Table	13.2.2.9.46
0x0020F06C + 0x4*n, n=0...2	GL_PREEXT_FLGS_L1SEL0_1[n]	L1 Flag Select Control (0-1)	13.2.2.9.47
0x0020F078 + 0x4*n, n=0...2	GL_PREEXT_FLGS_L1SEL2_3[n]	L1 Flag Select Control (2-3)	13.2.2.9.48
0x0020F084 + 0x4*n, n=0...2	GL_PREEXT_CTLTBL_L2ADDR[n]	L2 Configuration Table Address	13.2.2.9.49
0x0020F090 + 0x4*n, n=0...2	GL_PREEXT_CTLTBL_L2DATA[n]	L2 Configuration Table Data	13.2.2.9.50
0x0020F09C + 0x4*n, n=0...2	GL_PREEXT_L2PRTMOD[n]	XLT1, XLT2 Partition Mode	13.2.2.9.51
0x0020F0A8 + 0x4*n, n=0...2	GL_PREEXT_L2BMP0_3[n]	PG L2 CDID Bitmap (LSB)	13.2.2.9.52
0x0020F0B4 + 0x4*n, n=0...2	GL_PREEXT_L2BMP4_7[n]	PG L2 CDID Bitmap (MSB)	13.2.2.9.53
0x0020F0C0 + 0x4*n, n=0...2	GL_PREEXT_XLT1_L2ADDR[n]	XLT1 Table Configuration Address	13.2.2.9.54
0x0020F0CC + 0x4*n, n=0...2	GL_PREEXT_XLT1_L2DATA[n]	XLT1 Table Configuration Data	13.2.2.9.55
0x0020F0D8 + 0x4*n, n=0...2	GL_PREEXT_XLT2_L2ADDR[n]	XLT2 Table Configuration Address	13.2.2.9.56
0x0020F0E4 + 0x4*n, n=0...2	GL_PREEXT_XLT2_L2DATA[n]	XLT2 Table Configuration Data	13.2.2.9.57
0x0020F0F0 + 0x4*n, n=0...2	GL_PREEXT_PID_L2GKTYPE[n]	Profile ID Gen Key Type	13.2.2.9.58
0x0020F0FC + 0x4*n, n=0...2	GL_PREEXT_L2_PMASK0[n]	Profile Key Mask (LSB)	13.2.2.9.59
0x0020F108 + 0x4*n, n=0...2	GL_PREEXT_L2_PMASK1[n]	Profile Key Mask (MSB)	13.2.2.9.60
0x0020F114 + 0x4*n, n=0...2	GL_PREEXT_TCAM_L2ADDR[n]	TCAM Configuration (Address)	13.2.2.9.61
0x0020F120 + 0x4*n, n=0...2	GL_PREEXT_TCAM_L2DATALSB[n]	TCAM Configuration LSB (Data)	13.2.2.9.62
0x0020F12C + 0x4*n, n=0...2	GL_PREEXT_TCAM_L2DATAMSB[n]	TCAM Configuration MSB (Data+Mask)	13.2.2.9.63
0x0020F138 + 0x4*n, n=0...2	GL_PREEXT_DFLT_L2PRFL[n]	L2 Default Profile	13.2.2.9.64
0x0020F144 + 0x4*n, n=0...2	GL_PREEXT_K2N_L2ADDR[n]	K2N Table Configuration Address	13.2.2.9.65

Table 13-17. PF - FlexiPipe Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0020F150 + 0x4*n, n=0...2	GL_PREEXT_K2N_L2DATA[n]	K2N Table Configuration Data	13.2.2.9.66
0x0020F15C + 0x4*n, n=0...2	GL_PREEXT_N2N_L2ADDR[n]	K2N Table Configuration Address	13.2.2.9.67
0x0020F168 + 0x4*n, n=0...2	GL_PREEXT_N2N_L2DATA[n]	K2N Table Configuration Data	13.2.2.9.68
0x0020F498 + 0x4*n, n=0...2	GL_PREEXT_L2_TMASK0[n]	Profile Key Mask (LSB)	13.2.2.9.69
0x0020F4A4 + 0x4*n, n=0...2	GL_PREEXT_L2_TMASK1[n]	Profile Key Mask (MSB)	13.2.2.9.70
0x00210000 + 0x4*n, n=0...2	GL_ACLEXT_FORCE_PID[n]	Force Profile ID	13.2.2.9.71
0x0021000C + 0x4*n, n=0...2	GL_ACLEXT_PLVL_SEL[n]	Profile Level Selector	13.2.2.9.72
0x00210018 + 0x4*n, n=0...2	GL_ACLEXT_FORCE_L1CDID[n]	L1 Force CDID	13.2.2.9.73
0x00210024 + 0x4*n, n=0...2	GL_ACLEXT_P2P_L1ADDR[n]	L1 P2P Table Configuration Address	13.2.2.9.74
0x00210030 + 0x4*n, n=0...2	GL_ACLEXT_P2P_L1DATA[n]	L1 P2P Table Configuration Data	13.2.2.9.75
0x0021003C + 0x4*n, n=0...2	GL_ACLEXT_XLT0_L1ADDR[n]	XLT0 Table Configuration Address	13.2.2.9.76
0x00210048 + 0x4*n, n=0...2	GL_ACLEXT_XLT0_L1DATA[n]	XLT0 Table Configuration Data	13.2.2.9.77
0x00210054 + 0x4*n, n=0...2	GL_ACLEXT_CDMD_L1SEL[n]	L1 Bidirectional CTL	13.2.2.9.78
0x00210060 + 0x4*n, n=0...2	GL_ACLEXT_FLGS_L1TBL[n]	L1 Flag Select Table	13.2.2.9.79
0x0021006C + 0x4*n, n=0...2	GL_ACLEXT_FLGS_L1SEL0_1[n]	L1 Flag Select Control (0-1)	13.2.2.9.80
0x00210078 + 0x4*n, n=0...2	GL_ACLEXT_FLGS_L1SEL2_3[n]	L1 Flag Select Control (2-3)	13.2.2.9.81
0x00210084 + 0x4*n, n=0...2	GL_ACLEXT_CTLTBL_L2ADDR[n]	L2 Configuration Table Address	13.2.2.9.82
0x00210090 + 0x4*n, n=0...2	GL_ACLEXT_CTLTBL_L2DATA[n]	L2 Configuration Table Data	13.2.2.9.83
0x0021009C + 0x4*n, n=0...2	GL_ACLEXT_L2PRTMOD[n]	XLT1, XLT2 Partition Mode	13.2.2.9.84
0x002100A8 + 0x4*n, n=0...2	GL_ACLEXT_L2BMP0_3[n]	PG L2 CDID Bitmap (LSB)	13.2.2.9.85
0x002100B4 + 0x4*n, n=0...2	GL_ACLEXT_L2BMP4_7[n]	PG L2 CDID Bitmap (MSB)	13.2.2.9.86
0x002100C0 + 0x4*n, n=0...2	GL_ACLEXT_XLT1_L2ADDR[n]	XLT1 Table Configuration Address	13.2.2.9.87
0x002100CC + 0x4*n, n=0...2	GL_ACLEXT_XLT1_L2DATA[n]	XLT1 Table Configuration Data	13.2.2.9.88
0x002100D8 + 0x4*n, n=0...2	GL_ACLEXT_XLT2_L2ADDR[n]	XLT2 Table Configuration Address	13.2.2.9.89
0x002100E4 + 0x4*n, n=0...2	GL_ACLEXT_XLT2_L2DATA[n]	XLT2 Table Configuration Data	13.2.2.9.90
0x002100F0 + 0x4*n, n=0...2	GL_ACLEXT_PID_L2GKTYPE[n]	Profile ID Gen Key Type	13.2.2.9.91
0x002100FC + 0x4*n, n=0...2	GL_ACLEXT_L2_PMASK0[n]	Profile Key Mask (LSB)	13.2.2.9.92
0x00210108 + 0x4*n, n=0...2	GL_ACLEXT_L2_PMASK1[n]	Profile Key Mask (MSB)	13.2.2.9.93
0x00210114 + 0x4*n, n=0...2	GL_ACLEXT_TCAM_L2ADDR[n]	TCAM Configuration (Address)	13.2.2.9.94
0x00210120 + 0x4*n, n=0...2	GL_ACLEXT_TCAM_L2DATALSB[n]	TCAM Configuration LSB (Data)	13.2.2.9.95
0x0021012C + 0x4*n, n=0...2	GL_ACLEXT_TCAM_L2DATAMSB[n]	TCAM Configuration MSB (Data)	13.2.2.9.96
0x00210138 + 0x4*n, n=0...2	GL_ACLEXT_DFLT_L2PRFL[n]	L2 Default Profile	13.2.2.9.97
0x00210144 + 0x4*n, n=0...2	GL_ACLEXT_K2N_L2ADDR[n]	K2N Table Configuration Address	13.2.2.9.98
0x00210150 + 0x4*n, n=0...2	GL_ACLEXT_K2N_L2DATA[n]	K2N Table Configuration Data	13.2.2.9.99
0x0021015C + 0x4*n, n=0...2	GL_ACLEXT_N2N_L2ADDR[n]	K2N Table Configuration Address	13.2.2.9.100
0x00210168 + 0x4*n, n=0...2	GL_ACLEXT_N2N_L2DATA[n]	K2N Table Configuration Data	13.2.2.9.101
0x00210498 + 0x4*n, n=0...2	GL_ACLEXT_L2_TMASK0[n]	Profile Key Mask (LSB)	13.2.2.9.102

Table 13-17. PF - FlexiPipe Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x002104A4 + 0x4*n, n=0...2	GL_ACLEXT_L2_TMASK1[n]	Profile Key Mask (MSB)	13.2.2.9.103
0x00393800 + 0x4*n, n=0...2	GL_ACLEXT_DFLT_L2PRFL_ACL[n]	L2 Default Profile	13.2.2.9.104
0x00458000 + 0x4*n + 0x100*m, n=0...63, m=0...31	GLFLXP_TX_SCHED_CORRECT[n,m]	Tx Scheduling Correction Control	13.2.2.9.105
0x0045A000 + 0x4*n + 0x400*m, n=0...255, m=0...5	GLFLXP_RX_CMD_PROTIDS[n,m]	ProtIDs for Creating RRX CMD Offsets	13.2.2.9.106
0x0045C000 + 0x4*n, n=0...255	GLFLXP_PTYPE_TRANSLATION[n]	PTYPE_10b to PTYPE_8b Translation	13.2.2.9.107
0x0045C400 + 0x4*n, n=0...255	GLFLXP_RX_CMD_LX_PROT_IDX[n]	LX Prot & Index for Rx CMD	13.2.2.9.108
0x0045C800 + 0x4*n, n=0...63	GLFLXP_RXDID_FLX_WRD_0[n]	RXDID FlexiWord 0 Control	13.2.2.9.109
0x0045C900 + 0x4*n, n=0...63	GLFLXP_RXDID_FLX_WRD_1[n]	RXDID FlexiWord 1 Control	13.2.2.9.110
0x0045CA00 + 0x4*n, n=0...63	GLFLXP_RXDID_FLX_WRD_2[n]	RXDID FlexiWord 2 Control	13.2.2.9.111
0x0045CB00 + 0x4*n, n=0...63	GLFLXP_RXDID_FLX_WRD_3[n]	RXDID FlexiWord 3 Control	13.2.2.9.112
0x0045CC00 + 0x4*n, n=0...63	GLFLXP_RXDID_FLX_WRD_4[n]	RXDID FlexiWord 4 Control	13.2.2.9.113
0x0045CD00 + 0x4*n, n=0...63	GLFLXP_RXDID_FLX_WRD_5[n]	RXDID FlexiWord 5 Control	13.2.2.9.114
0x0045D000 + 0x4*n + 0x100*m, n=0...63, m=0...4	GLFLXP_RXDID_FLAGS[n,m]	RXDID FlexiFlags Control	13.2.2.9.115
0x0045D600 + 0x4*n, n=0...63	GLFLXP_RXDID_FLAGS1_OVERRIDE[n]	RXDID Flags1 Override Control	13.2.2.9.116
0x00480000 + 0x4*QRX, QRX=0...2047	QRXFLXP_CNTXT[QRX]	Queue Context Flex Extension	13.2.2.9.117

Table 13-18. PF - Parser Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00200004	GL_PRS_RX_SIZE_CTRL	PRS Balancer Config	13.2.2.10.1
0x0020000C + 0x4*n, n=0...6	GL_PRS_RX_PIPE_INIT0[n]	Rx-Query Pipe-Status Init for Word 0-6	13.2.2.10.2
0x00200028	GL_PRS_RX_PIPE_INIT1	Rx-Query Pipe-Status Init for Word 7	13.2.2.10.3
0x0020002C	GL_PRS_RX_PIPE_INIT2	Rx-Query Pipe-Status Init for Word 8	13.2.2.10.4
0x002001C0	GL_XLR_MARKER_TRIG_RCU_PRS	XLR Marker Trigger	13.2.2.10.5
0x002001C4 + 0x4*n, n=0...3	GL_QH_MARKER_TRIG_RCU_PRS[n]	QH Removal Marker Trigger	13.2.2.10.6
0x002001D4 + 0x4*n, n=0...7	GL_COTF_MARKER_TRIG_RCU_PRS[n]	COTF Marker Trigger	13.2.2.10.7
0x002001F4 + 0x4*n, n=0...1	GL_XLR_MARKER_STATUS[n]	XLR Debug Markers Status	13.2.2.10.8
0x002001FC	GL_QH_MARKER_STATUS	QH Debug Markers Status	13.2.2.10.9
0x00200200	GL_COTF_MARKER_STATUS	COTF Debug Markers Status	13.2.2.10.10
0x00200204	GL_PRS_MARKER_ERROR	PRS Markers Error Indication (Marker FIFO Full)	13.2.2.10.11
0x00200208 + 0x4*n, n=0...63	GL_XLR_MARKER_LOG_RCU_PRS[n]	PRS Marker FIFO Read Access	13.2.2.10.12
0x00200708	GL_RPRS_ANA_CSR_CTRL	Rx ANA CSR Access Control	13.2.2.10.13
0x00202000	GL_TPRS_PM_THR	Pipe Monitor Threshold	13.2.2.10.14

Table 13-18. PF - Parser Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00202004	GL_TPRS_MNG_PM_THR	MNG Pipe Monitor Threshold	13.2.2.10.15
0x00202008 + 0x4*n, n=0...1	GL_TPRS_PM_CNT[n]	Pipe Monitor Counters Status	13.2.2.10.16
0x00202014	GL_PRS_TX_SIZE_CTRL	Tx-Query Min/Max Size Control	13.2.2.10.17
0x00202018 + 0x4*n, n=0...6	GL_PRS_TX_PIPE_INIT0[n]	Tx-Query Pipe-Status Init for Word 0-6	13.2.2.10.18
0x00202034	GL_PRS_TX_PIPE_INIT1	Tx-Query Pipe-Status Init for Word 7	13.2.2.10.19
0x00202038	GL_PRS_TX_PIPE_INIT2	Tx-Query Pipe-Status Init for Word 8	13.2.2.10.20
0x00202100	GL_TPRS_ANA_CSR_CTRL	Tx ANA CSR Access Control	13.2.2.10.21
0x005008C0	GL_XLR_MARKER_TRIG_PE	XLR Marker Trigger PE	13.2.2.10.22

Table 13-19. PF - Switch Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00040840 + 0x4*n, n=0...31	PRT_TCTUPR[n]	Port - TC Transmit UP Replacement	13.2.2.11.1
0x001D2698	GL_SWT_FUNCFILT	IPsec Function Limiting	13.2.2.11.2
0x00204000	GL_SWT_LAT_SINGLE	Large Action - Single Action Offset	13.2.2.11.3
0x00204004	GL_SWT_LAT_DOUBLE	Large Action - Double Action Offset	13.2.2.11.4
0x00204008	GL_SWT_LAT_QUAD	Large Action - Quad Action Offset	13.2.2.11.5
0x0020401C	EMP_SWT_REPIND	Replication Table Control	13.2.2.11.6
0x00204020	EMP_SWT_PRUNIND	Prune Table Control	13.2.2.11.7
0x002040A4	GL_OVERRIDEC	Unallowed Override Attempt Count	13.2.2.11.8
0x002040AC	GL_SWT_MD_PRI	Switch Metadata Priority	13.2.2.11.9
0x00204100	PRT_SWT_MSCCNT	Storm Control - Multicast Current Count	13.2.2.11.10
0x00204120	PRT_SBPVSI	Port - Store Bad Packets VSI	13.2.2.11.11
0x00204140	PRT_SCSTS	Storm Control - Status	13.2.2.11.12
0x00204160	PRT_SWT_BSCCNT	Storm Control - Broadcast Current Count	13.2.2.11.13
0x00204180	PRT_SWT_BSCTRH	Storm Control - Broadcast Threshold	13.2.2.11.14
0x002041C0	PRT_SWT_MSCTRH	Storm Control - Multicast Threshold	13.2.2.11.15
0x002041E0	PRT_SWT_SCBI	Storm Control - Basic Interval	13.2.2.11.16
0x00204200	PRT_SWT_SCCRL	Storm Control - Control Register	13.2.2.11.17
0x00204280	PRT_SWT_MIRIG	Mirror - LAN Port Ingress Rule	13.2.2.11.18
0x002042A0	PRT_SWT_MIREG	Mirror - LAN Port Egress Rule	13.2.2.11.19
0x00204500 + 0x4*n, n=0...63	GL_SWT_MIRTARVSI[n]	Mirror - Target VSI	13.2.2.11.20
0x0020A1A4 + 0x4*n, n=0...255	GLSWID_STAT_BLOCK[n]	SWID Stat Block ID	13.2.2.11.21
0x0020A5A4	GLSWT_ACT_RESP_0	Switch Recipes Used	13.2.2.11.22
0x0020A5A8	GLSWT_ACT_RESP_1	Switch Recipes Used	13.2.2.11.23
0x0020A5AC	GL_PLG_AVG_CALC_CFG	Throughput Counters Config	13.2.2.11.24

Table 13-19. PF - Switch Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0020A5B0	GL_PLG_AVG_CALC_ST	Throughput Counters Status	13.2.2.11.25
0x0020A674	GLSWT_ARB_MODE	Hardware Arb Control	13.2.2.11.26
0x00214074 + 0x4*n, n=0...6	GL_PRE_CFG_DATA[n]	Recipe Data	13.2.2.11.27
0x00214090	GL_PRE_CFG_CMD	Recipe Command	13.2.2.11.28
0x00214094 + 0x4*n, n=0...31	GL_VP_SWITCHID[n]	Virtual Port Switch ID	13.2.2.11.29
0x00214114	GL_SWT_SWIDFVIDX	SWID Field Vector Index	13.2.2.11.30
0x00216000 + 0x4*n, n=0...5	GL_SWT_FW_STS[n]	FW Config Status	13.2.2.11.31

Table 13-20. PF - VSI Context Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00041000 + 0x4*VSI, VSI=0...767	VSI_TIR_0[VSI]	VSI Tag Insert Register - First Tag	13.2.2.12.1
0x00042000 + 0x4*VSI, VSI=0...767	VSI_TIR_1[VSI]	VSI Tag Insert Register - Second Tag	13.2.2.12.2
0x00043000 + 0x4*VSI, VSI=0...767	VSI_TIR_2[VSI]	VSI Tag Insert Register - Third Tag	13.2.2.12.3
0x00044000 + 0x4*VSI, VSI=0...767	VSI_TAIR[VSI]	VSI Tag Alternate Insert Register	13.2.2.12.4
0x00045000 + 0x4*VSI, VSI=0...767	VSI_TAR[VSI]	VSI Tag Accept Register	13.2.2.12.5
0x00046000 + 0x4*VSI, VSI=0...767	VSI_L2TAGSTXVALID[VSI]	VSI L2 Tx Tags Control	13.2.2.12.6
0x00047000 + 0x4*VSI, VSI=0...767	VSI_TUPR[VSI]	VSI Transmit UP Replacement	13.2.2.12.7
0x00048000 + 0x4*VSI, VSI=0...767	VSI_TUPIOM[VSI]	VSI Transmit UP Inner to Outer Mapping	13.2.2.12.8
0x00050000 + 0x4*VSI, VSI=0...767	VSI_RUPR[VSI]	VSI Receive UP Replacement	13.2.2.12.9
0x00051000 + 0x4*VSI, VSI=0...767	VSI_TSR[VSI]	VSI Tag Strip Register	13.2.2.12.10
0x0009C000 + 0x4*VSI, VSI=0...767	VSI_PASID[VSI]	PASID Context	13.2.2.12.11
0x001D0000 + 0x4*VSI, VSI=0...767	VSI_VSI2F[VSI]	VSI to Function Mapping Multicast	13.2.2.12.12
0x00205000 + 0x4*VSI, VSI=0...767	VSI_RXSWCTRL[VSI]	VSI Rx Switch Control	13.2.2.12.13
0x00207000 + 0x4*VSI, VSI=0...767	VSI_SWT_MIREG[VSI]	Mirror - Rx Rules VSIs	13.2.2.12.14
0x00208000 + 0x4*VSI, VSI=0...767	VSI_SWT_MIRIG[VSI]	Mirror - Tx Rules VSIs	13.2.2.12.15
0x00209000 + 0x4*VSI, VSI=0...767	VSI_SRCSWCTRL[VSI]	VSI Source Switch Control	13.2.2.12.16
0x00215000 + 0x4*VSI, VSI=0...767	VSI_SWITCHID[VSI]	Source VSI Switch ID	13.2.2.12.17

Table 13-20. PF - VSI Context Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00400000 + 0x1000*n + 0x4*VSI, n=0...12, VSI=0...767	VSIQF_HKEY[n,VSI]	VSI Classification Filter - Hash Key	13.2.2.12.18
0x0040D000 + 0x4*VSI, VSI=0...767	VSIQF_HASH_CTL[VSI]	VSI Classification Filter - Hash Control	13.2.2.12.19
0x00411000 + 0x4*VSI, VSI=0...767	VSIQF_FD_CTL1[VSI]	VSI Classification Filter - FD Control 1	13.2.2.12.20
0x00414000 + 0x4*VSI, VSI=0...767	VSIQF_PE_CTL1[VSI]	VSI Classification Filter - PE Control 1	13.2.2.12.21
0x00420000 + 0x1000*n + 0x4*VSI, n=0...15, VSI=0...767	VSIQF_HLUT[n,VSI]	VSI Classification Filter - Hash LUT	13.2.2.12.22
0x00448000 + 0x1000*n + 0x4*VSI, n=0...3, VSI=0...767	VSIQF_TC_REGION[n,VSI]	VSI Classification Filter - Receive TC Queue Regions	13.2.2.12.23
0x00457000 + 0x4*VSI, VSI=0...767	VSIQF_FD_DFLT[VSI]	VSI Classification Filter - FD Default Action	13.2.2.12.24
0x00462000 + 0x4*VSI, VSI=0...767	VSIQF_FD_SIZE[VSI]	VSI Classification Filter - FD VSI Space Sizes	13.2.2.12.25
0x00464000 + 0x4*VSI, VSI=0...767	VSIQF_FD_CNT[VSI]	VSI Classification Filter - FD VSI Space Counters	13.2.2.12.26

Table 13-21. PF - ACL Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00391000	GL_ACL_ACCESS_CMD	Configuration Access Command	13.2.2.13.1
0x00391004	GL_ACL_ACCESS_STATUS	Configuration Access Status	13.2.2.13.2
0x00391008 + 0x4*n, n=0...31	GL_ACL_PROFILE_BWSB_SEL[n]	Byte and Word Selection Bases Select per Profile	13.2.2.13.3
0x00391088 + 0x4*n, n=0...15	GL_ACL_PROFILE_DWSB_SEL[n]	DWord Selection Base Select per Profile	13.2.2.13.4
0x003910C8 + 0x4*n, n=0...7	GL_ACL_PROFILE_PF_CFG[n]	Profile Assignment to Scenario	13.2.2.13.5
0x003910E8 + 0x4*n, n=0...7	GL_ACL_PROFILE_RC_CFG[n]	Range Checker Configuration per Profile	13.2.2.13.6
0x00391108 + 0x4*n, n=0...7	GL_ACL_PROFILE_RCF_MASK[n]	Word Selection Base Range Checked Fields Masking per Profile	13.2.2.13.7
0x00391168 + 0x4*n, n=0...15	GL_ACL_DEFAULT_ACT[n]	Default Action Array	13.2.2.13.8
0x00391800 + 0x4*VSI, VSI=0...767	VSI_ACL_DEF_SEL[VSI]	VSI Dependent ACL Configuration	13.2.2.13.9
0x00393810	GL_ACL_CHICKEN_REGISTER	GL_ACL_CHICKEN_REGISTER	13.2.2.13.10
0x00393814	GL_ACL_TCAM_KEY_L	TCAM Write Key Low	13.2.2.13.11
0x00393818	GL_ACL_TCAM_KEY_H	TCAM Write Key High	13.2.2.13.12
0x0039381C	GL_ACL_TCAM_KEY_INV_L	TCAM Write Key Invert Low	13.2.2.13.13
0x00393820	GL_ACL_TCAM_KEY_INV_H	TCAM Write Key Invert High	13.2.2.13.14

Table 13-21. PF - ACL Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00393824 + 0x4*n, n=0...1	GL_ACL_ACTMEM_ACT[n]	Action Write Data	13.2.2.13.15
0x0039382C + 0x4*n, n=0...15	GL_ACL_SCENARIO_CFG_L[n]	Scenario Configuration Write Data Low Part	13.2.2.13.16
0x0039386C + 0x4*n, n=0...15	GL_ACL_SCENARIO_CFG_H[n]	Scenario Configuration Write Data High Part	13.2.2.13.17
0x003938AC + 0x4*n, n=0...19	GL_ACL_SCENARIO_ACT_CFG[n]	Scenario Action RAM Configuration Write Data	13.2.2.13.18

Table 13-22. PF - Rx Filters Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x000AA078 + 0x4*n, n=0...1	GLQF_PETABLE_CLR[n]	Global Classification Filter - PE Table Clear	13.2.2.14.1
0x0020E514	GLQF_PE_FVE	PF Classification Filter - PE Field Vector Bitmap Enable	13.2.2.14.2
0x0040E000 + 0x4*n + 0x200*m, n=0...127, m=0...5	GLQF_HINSET[n,m]	Global Classification Filter - Hash Input Set	13.2.2.14.3
0x0040F000 + 0x4*n + 0x200*m, n=0...127, m=0...5	GLQF_HSYMM[n,m]	Global Classification Filter - Symmetric Hash	13.2.2.14.4
0x0040FC00 + 0x4*n, n=0...31	GLQF_HMASK[n]	Global Classification Filter - Hash Mask	13.2.2.14.5
0x00410000 + 0x4*n, n=0...127	GLQF_HMASK_SEL[n]	Global Classification Filter - Hash Mask Select	13.2.2.14.6
0x00410400 + 0x4*n, n=0...127	GLQF_FDMASK_SEL[n]	Global Classification Filter - FD Mask Select	13.2.2.14.7
0x00410800 + 0x4*n, n=0...31	GLQF_FDMASK[n]	Global Classification Filter - FD Mask	13.2.2.14.8
0x00412000 + 0x4*n + 0x200*m, n=0...127, m=0...5	GLQF_FDINSET[n,m]	Global Classification Filter - FD Input Set	13.2.2.14.9
0x00413000 + 0x4*n + 0x200*m, n=0...127, m=0...5	GLQF_FDSWAP[n,m]	Global Classification Filter - FD SWAP	13.2.2.14.10
0x00415000 + 0x4*n + 0x80*m, n=0...31, m=0...5	GLQF_PEINSET[n,m]	Global Classification Filter - PE Input Set	13.2.2.14.11
0x00415400 + 0x4*n, n=0...15	GLQF_PEMASK[n]	Global Classification Filter - PE Mask	13.2.2.14.12
0x00415500 + 0x4*n, n=0...31	GLQF_PEMASK_SEL[n]	Global Classification Filter - PE Mask Select	13.2.2.14.13
0x00430000 + 0x40*n, n=0...511	PFQF_HLUT[n]	PF Classification Filter - Hash LUT	13.2.2.14.14
0x00438000 + 0x4*n + 0x200*m, n=0...127, m=0...15	GLQF_HLUT[n,m]	Global Classification Filter - Hash LUT	13.2.2.14.15
0x0043A000	PFQF_FD_ENA	PF Classification Filter - FD Enable	13.2.2.14.16
0x0043A080	PFQF_PE_FILTERING_ENA	PF Classification Filter - PE Enable	13.2.2.14.17

Table 13-22. PF - Rx Filters Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0044D000 + 0x4*n + 0x200*m, n=0...127, m=0...3	GLQF_PROF2TC[n,m]	Global Classification Filter - Packet Profile to Hash TC Region Mapping	13.2.2.14.18
0x00450000 + 0x4*n, n=0...2047	GLQF_APBVT[n]	Global Classification Filter Accelerated Port Bit Vector	13.2.2.14.19
0x00452000 + 0x4*n, n=0...3	GLQF_FDEVICTENA[n]	Global Classification Filter - FD Profile Evict Enable	13.2.2.14.20
0x00452080	PFQF_PE_TC_CTL	PF Classification Filter - QH TC Enable	13.2.2.14.21
0x00455200 + 0x4*n, n=0...31	GLQF_PE_CTL2[n]	Global Classification Filter - PE Control 2	13.2.2.14.22
0x00455400 + 0x4*n, n=0...15	GLQF_HLUT_SIZE[n]	Global Classification Filter - Hash LUT Size	13.2.2.14.23
0x00455480	PFQF_HLUT_SIZE	PF Classification Filter - Hash LUT Size	13.2.2.14.24
0x00455500	GLQF_PE_APBVT_CNT	Global PE APBVT LAN Packet Counter	13.2.2.14.25
0x00455800 + 0x4*VF, VF=0...255	VPQF_PE_FILTERING_ENA[VF]	VF Classification Filter - PE Enable	13.2.2.14.26
0x00456000 + 0x4*n, n=0...12	GLQF_HKEY[n]	Global Classification Filter - Hash Key	13.2.2.14.27
0x00460000	GLQF_FD_CTL	Global Classification Filter Control	13.2.2.14.28
0x00460010	GLQF_FD_SIZE	Global Classification Filter - FD Space Size	13.2.2.14.29
0x00460018	GLQF_FD_CNT	Global Classification Filter - FD Space Counters	13.2.2.14.30
0x00460100	PFQF_FD_SIZE	PF Classification Filter - FD Space Sizes	13.2.2.14.31
0x00460180	PFQF_FD_CNT	Global Classification Filter - FD PF Space Counter	13.2.2.14.32
0x00460200	PFQF_FD_SUBTRACT	Global Classification Filter - FD PF Space Counter	13.2.2.14.33
0x00470000	PFQF_PE_CTL1	PF Classification Filter - PE Control	13.2.2.14.34
0x00470040	PFQF_PE_CTL2	PF Classification Filter - PE Control	13.2.2.14.35
0x00470100	PFQF_PE_FLHD	PF Free List Head Array	13.2.2.14.36
0x00470200	PFQF_PECNT_0	PF Classification Filter PE Filter Counter 0	13.2.2.14.37
0x00470300	PFQF_PECNT_1	PF Classification Filter PE Filter Counter 1	13.2.2.14.38
0x00470400	PFQF_PE_ST_CTL	PF Control Register for the Statistic Counter	13.2.2.14.39
0x00470480	PFQF_PE_CLSN0	PF PE Classification Filter Collision Counter 0	13.2.2.14.40
0x00470500	PFQF_PE_CLSN1	PF PE Classification Filter Collision Counter 1	13.2.2.14.41
0x00471040	GLQF_PE_OSR_STS	Global PE Classification Filter Outstanding Request Counter	13.2.2.14.42
0x00471080	GLQF_PE_CMD	QH ADD/REM Commands Status	13.2.2.14.43
0x004710C0	GLQF_PE_CTL	Global Classification Filter Control	13.2.2.14.44
0x00472000 + 0x4*VF, VF=0...255	VPQF_PE_FLHD[VF]	VF Free List Head Array	13.2.2.14.45
0x00472800 + 0x4*VF, VF=0...255	VPQF_PECNT_0[VF]	VF Classification Filter PE Filter Counter 0	13.2.2.14.46
0x00473000 + 0x4*VF, VF=0...255	VPQF_PECNT_1[VF]	VF Classification Filter PE Filter Counter 1	13.2.2.14.47

Table 13-22. PF - Rx Filters Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00474000 + 0x4*VF, VF=0...255	VPQF_PE_CTL1[VF]	VF Classification Filter - PE Control	13.2.2.14.48
0x00474800 + 0x4*VF, VF=0...255	VPQF_PE_CTL2[VF]	PF Classification Filter - PE Control	13.2.2.14.49

Table 13-23. PF - Interrupt Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00088080	PFINT_GPIO_ENA	PF General Purpose IO Interrupt Enablement	13.2.2.15.1
0x000880C0	EMPINT_GPIO_ENA	EMP General Purpose IO Interrupt Enablement	13.2.2.15.2
0x0009D000 + 0x4*VF, VF=0...255	VPINT_ALLOC_PCI[VF]	VF Vector Allocation	13.2.2.15.3
0x0009D800	PFINT_ALLOC_PCI	PF Vector Allocation - PCI	13.2.2.15.4
0x00140000 + 0x4*DBQM, DBQM=0...16383	QINT_TQCTL[DBQM]	Transmit Queue Interrupt Cause Control	13.2.2.15.5
0x00150000 + 0x4*QRX, QRX=0...2047	QINT_RQCTL[QRX]	Receive Queue Interrupt Cause Control	13.2.2.15.6
0x00154000 + 0x2000*n + 0x4*INT, n=0...2, INT=0...2047	GLINT_ITR[n,INT]	Global Interrupt Throttling	13.2.2.15.7
0x0015A000 + 0x4*INT, INT=0...2047	GLINT_RATE[INT]	Global Interrupt Rate Limit	13.2.2.15.8
0x0015C000 + 0x4*INT, INT=0...2047	GLINT_CEQCTL[INT]	Global PE Completion Event Queue Interrupt Cause Control	13.2.2.15.9
0x00160000 + 0x4*INT, INT=0...2047	GLINT_DYN_CTL[INT]	Global Interrupt Dynamic Control	13.2.2.15.10
0x00162000 + 0x4*INT, INT=0...2047	GLINT_VECT2FUNC[INT]	Global Interrupt Vector 2 Function Allocation	13.2.2.15.11
0x0016A000 + 0x4*VSI, VSI=0...767	VPINT_MBX_CTL[VSI]	VF Mailbox Queue Mapping to Interrupt Control	13.2.2.15.12
0x0016B000 + 0x4*VP128, VP128=0...127	VPINT_MBX_CPM_CTL[VP128]	VF Mailbox Queue Mapping to Interrupt Control	13.2.2.15.13
0x0016B200 + 0x4*VP16, VP16=0...15	VPINT_MBX_HLP_CTL[VP16]	VF HLP Mailbox Queue Mapping to Interrupt Control	13.2.2.15.14
0x0016B240 + 0x4*VP16, VP16=0...15	VPINT_MBX_PSM_CTL[VP16]	VF PSM Mailbox Queue Mapping to Interrupt Control	13.2.2.15.15
0x0016B280	PFINT_MBX_CTL	PF Mailbox Queue Mapping to Interrupt Control	13.2.2.15.16
0x0016B2C0	PF0INT_MBX_CPM_CTL	PF0 CPM Mailbox Queue Mapping to Interrupt Control	13.2.2.15.17
0x0016B2C4	PF0INT_MBX_HLP_CTL	PF0 HLP Mailbox Queue Mapping to Interrupt Control	13.2.2.15.18
0x0016B2C8	PF0INT_MBX_PSM_CTL	PF0 PSM Mailbox Queue Mapping to Interrupt Control	13.2.2.15.19
0x0016B2CC	PF0INT_SB_CPM_CTL	PF0 CPM SB Queue Mapping to Interrupt Control	13.2.2.15.20
0x0016B400 + 0x4*VP128, VP128=0...127	VPINT_SB_CPM_CTL[VP128]	VF CPM SB Queue Mapping to Interrupt Control	13.2.2.15.21

Table 13-23. PF - Interrupt Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0016B600	PFINT_SB_CTL	PF SB Queue Mapping to Interrupt Control	13.2.2.15.22
0x0016B640	PF0INT_SB_HLP_CTL	PF0 HLPSB Queue Mapping to Interrupt Control	13.2.2.15.23
0x0016B800 + 0x4*VF, VF=0...255	VPINT_AEQCTL[VF]	VF PE Asynchronous Event Queue Interrupt Cause Control	13.2.2.15.24
0x0016C800	PFINT_FW_CTL	PF Firmware Admin Queue Mapping to Interrupt Control	13.2.2.15.25
0x0016C840	GLINT_FW_TOOL_CTL	Global Tools Firmware Admin Queue Mapping to Interrupt Control	13.2.2.15.26
0x0016C844	PF0INT_FW_HLP_CTL	PF0 HLP Firmware Admin Queue Mapping to Interrupt Control	13.2.2.15.27
0x0016C848	PF0INT_FW_PSM_CTL	PF0 PSM Firmware Admin Queue Mapping to Interrupt Control	13.2.2.15.28
0x0016C900	PFINT_OICR_ENA	PF Interrupt Other Cause Enablement	13.2.2.15.29
0x0016C980	PFINT_TSYN_MSK	Global Interrupt TimeSync PHY Mask	13.2.2.15.30
0x0016CA00	PFINT_OICR	PF Interrupt Other Cause	13.2.2.15.31
0x0016CA80	PFINT_OICR_CTL	PF Interrupt Other Cause Control	13.2.2.15.32
0x0016CB00	PFINT_AEQCTL	PF PE Asynchronous Event Queue Interrupt Cause Control	13.2.2.15.33
0x0016CC40	PF0INT_OICR_CPM	PF0 Interrupt Other Cause CPM	13.2.2.15.34
0x0016CC44	PF0INT_OICR_PSM	PF0 Interrupt Other Cause PSM	13.2.2.15.35
0x0016CC48	PF0INT_OICR_CTL_CPM	PF0 Interrupt Other Cause CPM Control	13.2.2.15.36
0x0016CC4C	PF0INT_OICR_ENA_HLP	PF0 Interrupt Other Cause HLP Enablement	13.2.2.15.37
0x0016CC50	GLINT_TSYN_PHY	Global Interrupt TimeSync PHY Indication	13.2.2.15.38
0x0016CC54	GLINT_CTL	Global Interrupt Control	13.2.2.15.39
0x0016CC58	PF0INT_OICR_ENA_PSM	PF0 Interrupt Other Cause PSM Enablement	13.2.2.15.40
0x0016CC5C	PF0INT_OICR_CTL_HLP	PF0 Interrupt Other Cause HLP Control	13.2.2.15.41
0x0016CC60	PF0INT_OICR_ENA_CPM	PF0 Interrupt Other Cause CPM Enablement	13.2.2.15.42
0x0016CC64	PF0INT_OICR_CTL_PSM	PF0 Interrupt Other Cause PSM Control	13.2.2.15.43
0x0016CC68	PF0INT_OICR_HLP	PF0 Interrupt Other Cause HLP	13.2.2.15.44
0x0016CCC0 + 0x4*n, n=0...1	GLINT_TSYN_PFMSTR[n]	Global Interrupt TimeSync Master Select	13.2.2.15.45
0x001D1000 + 0x4*VF, VF=0...255	VPINT_ALLOC[VF]	VF Vector Allocation	13.2.2.15.46
0x001D2600	PFINT_ALLOC	PF Vector Allocation	13.2.2.15.47

Table 13-24. PF - Virtualization PF Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0009DD80	PF_VT_PFALLOC_HIF	PF Resources Allocation	13.2.2.16.1
0x0009E680	PF_VIRT_VSTATUS	PF Virtualization Status Register	13.2.2.16.2

Table 13-24. PF - Virtualization PF Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x000BE080	PF_VT_PFALLOC_PCIE	PF Resources Allocation	13.2.2.16.3
0x001D2480	PF_VT_PFALLOC	PF Resources Allocation	13.2.2.16.4

Table 13-25. PF - DCB Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00040940	PRTDCB_TDPUC	DCB TDPU Control	13.2.2.17.1
0x00040960	PRTDCB_RUP_TDPU	DCB Receive UP in TDPU	13.2.2.17.2
0x00040980	PRTDCB_TX_DSCP2UP_CTL	Tx DCB DSCP to User Priority Control	13.2.2.17.3
0x000409A0 + 0x20*n, n=0...7	PRTDCB_TX_DSCP2UP_IPV4_LUT[n]	Tx DCB DSCP to User Priority LUT for IPv4 Packets	13.2.2.17.4
0x00040AA0 + 0x20*n, n=0...7	PRTDCB_TX_DSCP2UP_IPV6_LUT[n]	Tx DCB DSCP to User Priority LUT for IPv6 Packets	13.2.2.17.5
0x00049018 + 0x4*n, n=0...63	GL_DCB_TDSCP2TC_BLOCK_IPV4[n]	Transit DCSP to TC Enforcement - IPv4	13.2.2.17.6
0x00049118 + 0x4*n, n=0...63	GL_DCB_TDSCP2TC_BLOCK_IPV6[n]	Transit DCSP to TC Enforcement - IPv6	13.2.2.17.7
0x00049218	GL_DCB_TDSCP2TC_BLOCK_DIS	Transit DCSP to TC Enable	13.2.2.17.8
0x00083000	PRTDCB_GENC	Port DCB General Control	13.2.2.17.9
0x00083020	PRTDCB_GENS	Port DCB General Status	13.2.2.17.10
0x00083044	GLDCB_GENC	Global DCB General Control	13.2.2.17.11
0x000991C0	TPB_PRTDCB_TCB_DWRR_CREDITS	DCB Transmit Port DWRR Status	13.2.2.17.12
0x00099220	TPB_PRTDCB_TCB_DWRR_QUANTA	DCB Transmit Port DWRR Quanta/Weights	13.2.2.17.13
0x00099260	TPB_PRTDCB_TCB_DWRR_SAT	DCB Transmit Port DWRR Saturation Value	13.2.2.17.14
0x000992A0	TPB_PRTTCB_BULK_DWRR_REG_CREDITS	DCB Transmit Regular Bulk DWRR Status	13.2.2.17.15
0x000992C0	TPB_PRTTCB_BULK_DWRR_WB_CREDITS	DCB Transmit Wait Bulk DWRR Status	13.2.2.17.16
0x00099300	TPB_PRTTCB_LL_DWRR_REG_CREDITS	DCB Transmit Regular Low Latency DWRR Status	13.2.2.17.17
0x00099320	TPB_PRTTCB_LL_DWRR_WB_CREDITS	DCB Transmit Wait Low Latency DWRR Status	13.2.2.17.18
0x00099340	TPB_BULK_DWRR_REG_QUANTA	DCB Transmit Regular Bulk DWRR Quanta/ Weights	13.2.2.17.19
0x00099344	TPB_BULK_DWRR_WB_QUANTA	DCB Transmit Wait Bulk DWRR Quanta/ Weights	13.2.2.17.20
0x00099348	TPB_LL_DWRR_REG_QUANTA	DCB Transmit Regular Low Latency DWRR Quanta/Weights	13.2.2.17.21
0x0009934C	TPB_LL_DWRR_WB_QUANTA	DCB Transmit Wait Low Latency DWRR Quanta/Weights	13.2.2.17.22
0x00099350	TPB_BULK_DWRR_REG_SAT	DCB Transmit Regular Bulk DWRR Saturation Value	13.2.2.17.23
0x00099354	TPB_BULK_DWRR_WB_SAT	DCB Transmit Wait Bulk DWRR Saturation Value	13.2.2.17.24

Table 13-25. PF - DCB Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00099358	TPB_LL_DWRR_REG_SAT	DCB Transmit Regular Low Latency DWRR Saturation Value	13.2.2.17.25
0x0009935C	TPB_LL_DWRR_WB_SAT	DCB Transmit Wait Low Latency DWRR Saturation Value	13.2.2.17.26
0x00099360 + 0x4*n, n=0...31	TPB_WB_RL_TC_CFG[n]	DCB Transmit Rate Limiter Control per TC	13.2.2.17.27
0x000993E0 + 0x4*n, n=0...31	TPB_WB_RL_TC_STAT[n]	DCB Transmit Rate Limiter Status per TC	13.2.2.17.28
0x00099460	GLTPB_WB_RL	TC Rate Limiters Config	13.2.2.17.29
0x00099464	GLDCB_TPB_TCLL_CFG	TPB TC LL Config	13.2.2.17.30
0x00099468	GLDCB_TPB_IMM_TLPM	TPB TLPM TC Immediate FC Enable	13.2.2.17.31
0x0009946C	GLDCB_TPB_IMM_TPB	TPB TC Immediate FC Enable	13.2.2.17.32
0x0009949C	GLDCB_TFPFCI	Ignore FC per TC List	13.2.2.17.33
0x00099644	TPB_PRTTCB_CREDIT_EXP	TCB Arbiter Credit Expansion	13.2.2.17.34
0x00099664	TPB_GLTCB_CREDIT_EXP_CTL	TCB Arbiter Credit Expansion Control	13.2.2.17.35
0x0009966C	TPB_GLDCB_TCB_WB_SP	Global Wait Buffer Strict Priority Enable	13.2.2.17.36
0x000A0000	PRTDCB_TLPM_REG_DM	DCB Transmit Data Pipe Port Monitor Status	13.2.2.17.37
0x000A0020	PRTDCB_TLPM_REG_DTHR	DCB Transmit Data Pipe Port Monitor Threshold	13.2.2.17.38
0x000A0040	PRTDCB_TLPM_WAIT_PFC_DM	DCB Transmit Data Pipe Port Waiting Monitor Status	13.2.2.17.39
0x000A0060	PRTDCB_TLPM_WAIT_PFC_DTHR	DCB Transmit Data Pipe Port Waiting Monitor Threshold	13.2.2.17.40
0x000A0080 + 0x4*n, n=0...31	TCDCB_TLPM_WAIT_DM[n]	DCB Transmit Data Pipe TC Waiting Monitor Status	13.2.2.17.41
0x000A0100 + 0x4*n, n=0...31	TCDCB_TLPM_WAIT_DTHR[n]	DCB Transmit Data Pipe TC Waiting Monitor Threshold	13.2.2.17.42
0x000A0180	GLDCB_TLPM_PCI_DM	DCB PCIe Tx Data Count	13.2.2.17.43
0x000A0184	GLDCB_TLPM_PCI_DTHR	DCB PCIe Tx Data Threshold	13.2.2.17.44
0x000A018C	GLDCB_TLPM_IMM_TCUPM	DCB TC Immediate FC Enable	13.2.2.17.45
0x000A0190	GLDCB_TLPM_IMM_TCB	DCB TC Immediate FC Mode	13.2.2.17.46
0x000AE000	PRTDCB_TCB_DWRR_CREDITS	DCB Transmit Port DWRR Status	13.2.2.17.47
0x000AE020	PRTDCB_TCB_DWRR_QUANTA	DCB Transmit Port DWRR Quanta/Weights	13.2.2.17.48
0x000AE040	PRTDCB_TCB_DWRR_SAT	DCB Transmit Port DWRR Saturation Value	13.2.2.17.49
0x000AE060	PRTTCB_BULK_DWRR_REG_CREDITS	DCB Transmit Regular Bulk DWRR Status	13.2.2.17.50
0x000AE080	PRTTCB_BULK_DWRR_WB_CREDITS	DCB Transmit Wait Bulk DWRR Status	13.2.2.17.51
0x000AE0A0	PRTTCB_LL_DWRR_REG_CREDITS	DCB Transmit Regular Low Latency DWRR Status	13.2.2.17.52
0x000AE0C0	PRTTCB_LL_DWRR_WB_CREDITS	DCB Transmit Wait Low Latency DWRR Status	13.2.2.17.53
0x000AE0E0	GLTCB_BULK_DWRR_REG_QUANTA	DCB Transmit Regular Bulk DWRR Quanta/ Weights	13.2.2.17.54

Table 13-25. PF - DCB Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x000AE0E4	GLTCB_BULK_DWRR_WB_QUANTA	DCB Transmit Wait Bulk DWRR Quanta/ Weights	13.2.2.17.55
0x000AE0E8	GLTCB_LL_DWRR_REG_QUANTA	DCB Transmit Regular Low Latency DWRR Quanta/Weights	13.2.2.17.56
0x000AE0EC	GLTCB_LL_DWRR_WB_QUANTA	DCB Transmit Wait Low Latency DWRR Quanta/Weights	13.2.2.17.57
0x000AE0F0	GLTCB_BULK_DWRR_REG_SAT	DCB Transmit Regular Bulk DWRR Saturation Value	13.2.2.17.58
0x000AE0F4	GLTCB_BULK_DWRR_WB_SAT	DCB Transmit Wait Bulk DWRR Saturation Value	13.2.2.17.59
0x000AE0F8	GLTCB_LL_DWRR_REG_SAT	DCB Transmit Regular Low Latency DWRR Saturation Value	13.2.2.17.60
0x000AE0FC	GLTCB_LL_DWRR_WB_SAT	DCB Transmit Wait Low Latency DWRR Saturation Value	13.2.2.17.61
0x000AE100	PRTTCB_CREDIT_EXP	TCB Arbiter Credit Expansion	13.2.2.17.62
0x000AE120	GLTCB_CREDIT_EXP_CTL	TCB Arbiter Credit Expansion Control	13.2.2.17.63
0x000AE12C	GLDCB_TCB_MNG_SP	Global MNG LL Strict Priority Enable	13.2.2.17.64
0x000AE134	GLDCB_TCB_TCLL_CFG	TC Low Latency Config	13.2.2.17.65
0x000AE138 + 0x4*n, n=0...31	TCTCB_WB_RL_TC_CFG[n]	DCB Transmit Rate Limiter Control per TC	13.2.2.17.66
0x000AE1B8 + 0x4*n, n=0...31	TCTCB_WB_RL_TC_STAT[n]	DCB Transmit Rate Limiter Status per TC	13.2.2.17.67
0x000AE238	GLTCB_WB_RL	TC Rate Limiters Config	13.2.2.17.68
0x000AE310	GLDCB_TCB_WB_SP	Global Wait Buffer Strict Priority Enable	13.2.2.17.69
0x000BC360	PRTDCB_TCUPM_REG_CM	DCB Transmit Command Pipe Port Monitor Status	13.2.2.17.70
0x000BC380	PRTDCB_TCUPM_REG_CTHR	DCB Transmit Command Pipe Port Monitor Threshold	13.2.2.17.71
0x000BC3A0	PRTDCB_TCUPM_REG_DM	DCB Transmit Data Pipe Port Monitor Status	13.2.2.17.72
0x000BC3C0	PRTDCB_TCUPM_NO_EXCEED_DM	DCB Transmit Data non-Exceed Pipe Monitor Status	13.2.2.17.73
0x000BC3E0	PRTDCB_TCUPM_REG_DTHR	DCB Transmit Data Pipe Port Monitor Threshold	13.2.2.17.74
0x000BC400	PRTDCB_TCUPM_REG_PE_HB_DM	DCB Transmit Data Pipe Port Monitor Status	13.2.2.17.75
0x000BC420	PRTDCB_TCUPM_REG_PE_HB_DTHR	DCB Transmit Data Pipe Port Monitor Threshold	13.2.2.17.76
0x000BC440	PRTDCB_TCUPM_WAIT_PFC_CM	DCB Transmit Command Pipe Port Waiting Monitor Status	13.2.2.17.77
0x000BC460	PRTDCB_TCUPM_WAIT_PFC_CTHR	DCB Transmit Command Pipe Port Waiting Monitor Threshold	13.2.2.17.78
0x000BC480	PRTDCB_TCUPM_WAIT_PFC_DM	DCB Transmit Data Pipe Port Waiting Monitor Status	13.2.2.17.79
0x000BC4A0	PRTDCB_TCUPM_WAIT_PFC_DTHR	DCB Transmit Data Pipe Port Waiting Monitor Threshold	13.2.2.17.80

Table 13-25. PF - DCB Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x000BC4C0	PRTDCB_TCUPM_WAIT_PFC_PE_HB_DM	DCB Transmit Data Pipe Port Waiting Monitor Status	13.2.2.17.81
0x000BC4E0	PRTDCB_TCUPM_WAIT_PFC_PE_HB_DTHR	DCB Transmit Data Pipe Port Waiting Monitor Threshold	13.2.2.17.82
0x000BC520 + 0x4*n, n=0...31	TCDCB_TCUPM_WAIT_CM[n]	DCB Transmit Command Pipe TC Waiting Monitor Status	13.2.2.17.83
0x000BC5A0 + 0x4*n, n=0...31	TCDCB_TCUPM_WAIT_CTHR[n]	DCB Transmit Command Pipe TC Waiting Monitor Threshold	13.2.2.17.84
0x000BC620 + 0x4*n, n=0...31	TCDCB_TCUPM_WAIT_DM[n]	DCB Transmit Data Pipe TC Waiting Monitor Status	13.2.2.17.85
0x000BC6A0 + 0x4*n, n=0...31	TCDCB_TCUPM_WAIT_DTHR[n]	DCB Transmit Data Pipe TC Waiting Monitor Threshold	13.2.2.17.86
0x000BC720 + 0x4*n, n=0...31	TCDCB_TCUPM_WAIT_PE_HB_DM[n]	DCB Transmit Data Pipe TC Waiting Monitor Status	13.2.2.17.87
0x000BC7A0 + 0x4*n, n=0...31	TCDCB_TCUPM_WAIT_PE_HB_DTHR[n]	DCB Transmit Data Pipe TC Waiting Monitor Threshold	13.2.2.17.88
0x000BC824	GLDCB_TCUPM_IMM_EN	DCB TC Immediate FC Mode	13.2.2.17.89
0x000BC828	GLDCB_TCUPM_LEGACY_TC	DCB TC Legacy Queues Mapping	13.2.2.17.90
0x000BC830	GLDCB_TCUPM_NO_EXCEED_DIS	DCB Transmit Data non-Exceed Monitor Enable	13.2.2.17.91
0x000BC834	GLDCB_TCUPM_WB_DIS	DCB Transmit Wait Port Data Monitor Enable	13.2.2.17.92
0x001220C0	PRTDCB_RPRRC	DCB Receive Port Round Robin Control	13.2.2.17.93
0x001220E0	PRTDCB_RPRRS	DCB Receive Port Round Robin Status	13.2.2.17.94
0x00122100	GLDCB_RTC2PFC_RCB	QRX	13.2.2.17.95
0x00122140 + 0x4*n, n=0...31	GLDCB_RETSTCC[n]	DCB Receive ETS per TC Control	13.2.2.17.96
0x001221C0 + 0x4*n, n=0...31	GLDCB_RETSTCS[n]	DCB Receive ETS per TC Status	13.2.2.17.97
0x001222A0	PRTDCB_RETSC	DCB Receive ETS Control	13.2.2.17.98
0x001D2640	PRTDCB_RUP2TC	DCB Receive UP to TC Mapping	13.2.2.17.99
0x001D2694	GLDCB_TC2PFC	DCB TC to PFC Mapping	13.2.2.17.100
0x001D26C0	PRTDCB_TUP2TC	DCB Transmit UP to TC Mapping	13.2.2.17.101
0x001E4560	PRTDCB_TFCS	Transmit Flow Control Status	13.2.2.17.102
0x001E4580 + 0x20*n, n=0...3	PRTDCB_FCTTVN[n]	Flow Control Transmit Timer Value n	13.2.2.17.103
0x001E4600	PRTDCB_FCRTV	Flow Control Refresh Threshold Value	13.2.2.17.104
0x001E4640	PRTDCB_FCCFG	Flow Control Configuration	13.2.2.17.105
0x001E4660 + 0x20*n, n=0...7	PRTDCB_TPFCTS[n]	DCB Transmit PFC Timer Status	13.2.2.17.106
0x002000B0 + 0x4*n, n=0...31	GLDCB_PRS_RETSTCC[n]	DCB Receive ETS per TC Control	13.2.2.17.107
0x00200160	GLDCB_PRS_RSPMC	DCB Receive Shared Pipe Monitor Control	13.2.2.17.108
0x00200180	PRTDCB_PRS_RPRRC	DCB Receive Port Round Robin Control	13.2.2.17.109

Table 13-25. PF - DCB Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x002001A0	PRTDCB_PRS_RETSC	DCB Receive ETS Control	13.2.2.17.110
0x0020A040 + 0x4*n, n=0...31	GLDCB_SWT_RETSTCC[n]	DCB Receive ETS per TC Control	13.2.2.17.111
0x0020A140	PRTDCB_SWT_RETSC	DCB Receive ETS Control	13.2.2.17.112

Table 13-26. PF - Receive Packet Buffer Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x000AC000 + 0x4*n, n=0...15	GLRPB_DHW[n]	RPB Dedicated Pool High Watermark	13.2.2.18.1
0x000AC044 + 0x4*n, n=0...15	GLRPB_DLW[n]	RPB Dedicated Pool Low Watermark	13.2.2.18.2
0x000AC084 + 0x4*n, n=0...15	GLRPB_DPS[n]	RPB Dedicated Pool Size	13.2.2.18.3
0x000AC0C4 + 0x4*n, n=0...7	GLRPB_SPS[n]	RPB Shared Pool Size	13.2.2.18.4
0x000AC120 + 0x4*n, n=0...7	GLRPB_SHW[n]	RPB Shared Pool High Watermark	13.2.2.18.5
0x000AC140 + 0x4*n, n=0...7	GLRPB_SLW[n]	RPB Shared Pool Low Watermark	13.2.2.18.6
0x000AC2A4 + 0x4*n, n=0...31	GLRPB_TC_CFG[n]	TC Pool Config	13.2.2.18.7
0x000AC324	GLRPB_DSI_EN	DSI Traffic Enable	13.2.2.18.8
0x000AC330 + 0x4*n, n=0...31	GLRPB_TCHW[n]	RPB TC High Watermark	13.2.2.18.9
0x000AC3B0 + 0x4*n, n=0...31	GLRPB_TCLW[n]	RPB TC Low Watermark	13.2.2.18.10

Table 13-27. PF - Transmit Scheduler Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0051E228	GLPE_TSCD_PEPM	TSCD PEPM	13.2.2.19.1
0x0051E24C + 0x4*n, n=0...3	GLPE_TSCD_FLR[n]	Transmit Scheduler FLR	13.2.2.19.2
0x0051E2FC	GLPE_TSCD_NUM_PQS	Transmit Scheduler Number of PQs	13.2.2.19.3

Table 13-28. PF - Host Memory Cache Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x000AA074	GLFOC_CACHESIZE	FOC Cache Attributes	13.2.2.20.1
0x00100000	PFHMC_SDCMD_FPMAT	Private Memory Space Segment Descriptor Command	13.2.2.20.2
0x00100100	PFHMC_SDDATALOW_FPMAT	Private Memory Space Segment Descriptor Data Low	13.2.2.20.3
0x00100200	PFHMC_SDDATAHIGH_FPMAT	Private Memory Space Segment Descriptor Data High	13.2.2.20.4
0x00100300	PFHMC_PDINV_FPMAT	Private Memory Space Page Descriptor Invalidate	13.2.2.20.5
0x00100400	PFHMC_ERRORINFO_FPMAT	Host Memory Cache Error Information Register	13.2.2.20.6
0x00100500	PFHMC_ERRORDATA_FPMAT	Host Memory Cache Error Data Register	13.2.2.20.7

Table 13-28. PF - Host Memory Cache Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00100800 + 0x4*n, n=0...7	GLHMC_SDPART_FPMAT[n]	Private Memory Segment Table Partitioning Registers	13.2.2.20.8
0x00100880 + 0x4*n, n=0...7	GLHMC_PFPESDPART_FPMAT[n]	Private Memory Segment Table Partitioning Registers	13.2.2.20.9
0x0010202C	GLHMC_PEHTEOBJSZ_FPMAT	Private Memory PE Hash Table Entry Object Size	13.2.2.20.10
0x00102030	GLHMC_PEHTMAX_FPMAT	Private Memory Protocol Engine Hash Entry Max	13.2.2.20.11
0x00102074	GLHMC_FWSDDATALOW_FPMAT	Private Memory Space Segment Descriptor Data Low	13.2.2.20.12
0x00102078	GLHMC_FWSDDATAHIGH_FPMAT	Private Memory Space Segment Descriptor Data High	13.2.2.20.13
0x0010207C	GLHMC_FWPDINV_FPMAT	Private Memory Space Page Descriptor Invalidate	13.2.2.20.14
0x00104600 + 0x4*n, n=0...7	GLHMC_PEHTEBASE_FPMAT[n]	FPM PE Hash Table Entry Base	13.2.2.20.15
0x00104700 + 0x4*n, n=0...7	GLHMC_PEHTCNT_FPMAT[n]	FPM PE Hash Table Object Count	13.2.2.20.16
0x00108100 + 0x4*n, n=0...31	GLHMC_VFSDDATALOW_FPMAT[n]	Private Memory Space VF Segment Descriptor Data Low	13.2.2.20.17
0x00108200 + 0x4*n, n=0...31	GLHMC_VFSDDATAHIGH_FPMAT[n]	Private Memory Space VF Segment Descriptor Data High	13.2.2.20.18
0x00108300 + 0x4*n, n=0...31	GLHMC_VFPDINV_FPMAT[n]	Private Memory Space Page Descriptor Invalidate	13.2.2.20.19
0x00108800 + 0x4*n, n=0...31	GLHMC_VFSDPART_FPMAT[n]	Private Memory Segment Table Partitioning Registers	13.2.2.20.20
0x0010C600 + 0x4*n, n=0...31	GLHMC_VFPEHTEBASE_FPMAT[n]	FPM PE Hash Table Entry Base	13.2.2.20.21
0x0010C700 + 0x4*n, n=0...31	GLHMC_VFPEHTCNT_FPMAT[n]	FPM PE Hash Table Object Count	13.2.2.20.22
0x00110088	GLPDOC_CACHESIZE_FPMAT	PDOC Cache Attributes	13.2.2.20.23
0x00502E00 + 0x4*n, n=0...31	GLHMC_VFDBCQPART[n]	Private Memory CQ Doorbell Partition Registers	13.2.2.20.24
0x00502F00 + 0x4*n, n=0...31	GLHMC_VFCEQPART[n]	Private Memory CEQ Partitioning Registers	13.2.2.20.25
0x00503180 + 0x4*n, n=0...7	GLHMC_DBCQPART[n]	Private Memory CQ Doorbell Partition Registers	13.2.2.20.26
0x005031C0 + 0x4*n, n=0...7	GLHMC_CEQPART[n]	Private Memory CEQ Partitioning Registers	13.2.2.20.27
0x005044C0 + 0x4*n, n=0...7	GLHMC_DBQPPART[n]	Private Memory QP Doorbell Partition Registers	13.2.2.20.28
0x00504520 + 0x4*n, n=0...31	GLHMC_VFDBQPPART[n]	Private Memory VF QP Doorbell Partition Registers	13.2.2.20.29
0x005140A8	GLPEOC0_CACHESIZE	PEOC0 Cache Attributes	13.2.2.20.30
0x005160A8	GLPEOC1_CACHESIZE	PEOC1 Cache Attributes	13.2.2.20.31
0x00518074	GLPBLOC0_CACHESIZE	PBLOC0 Cache Attributes	13.2.2.20.32
0x0051A074	GLPBLOC1_CACHESIZE	PBLOC1 Cache Attributes	13.2.2.20.33
0x0051C06C	GLMDOC_CACHESIZE	MDOC Cache Attributes	13.2.2.20.34

Table 13-28. PF - Host Memory Cache Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00520000	PFHMC_SDCMD	Private Memory Space Segment Descriptor Command	13.2.2.20.35
0x00520100	PFHMC_SDDATALOW	Private Memory Space Segment Descriptor Data Low	13.2.2.20.36
0x00520200	PFHMC_SDDATAHIGH	Private Memory Space Segment Descriptor Data High	13.2.2.20.37
0x00520300	PFHMC_PDINV	Private Memory Space Page Descriptor Invalidate	13.2.2.20.38
0x00520400	PFHMC_ERRORINFO	Host Memory Cache Error Information Register	13.2.2.20.39
0x00520500	PFHMC_ERRORDATA	Host Memory Cache Error Data Register	13.2.2.20.40
0x00520800 + 0x4*n, n=0...7	GLHMC_SDPART[n]	Private Memory Segment Table Partitioning Registers	13.2.2.20.41
0x00520880 + 0x4*n, n=0...7	GLHMC_PFPESDPART[n]	Private Memory Segment Table Partitioning Registers	13.2.2.20.42
0x00522004	GLHMC_PEHDROBJSZ	Private Memory Protocol Engine Header Max	13.2.2.20.43
0x00522008	GLHMC_PEHDRMAX	FPM PE Header Object Count	13.2.2.20.44
0x0052200C	GLHMC_PEMDOBJSZ	Private Memory PE Metadata Object Size	13.2.2.20.45
0x00522010	GLHMC_PEMDMAX	Private Memory Protocol Engine Metadata Max	13.2.2.20.46
0x00522014	GLHMC_PEOOISCOBJSZ	Private Memory PE Out of Order Send Completion Object Size	13.2.2.20.47
0x00522018	GLHMC_PEOOISCMAX	Private Memory Protocol Engine Out of Order Send Completion Max	13.2.2.20.48
0x0052201C	GLHMC_PEQPOBJSZ	Private Memory PE QP Object Size	13.2.2.20.49
0x00522020	GLHMC_PECQOBJSZ	Private Memory PE CQ Object Size	13.2.2.20.50
0x0052202C	GLHMC_PEHTEOBJSZ	Private Memory PE Hash Table Entry Object Size	13.2.2.20.51
0x00522030	GLHMC_PEHTMAX	Private Memory Protocol Engine Hash Entry Max	13.2.2.20.52
0x00522034	GLHMC_PEARPOBJSZ	Private Memory PE ARP Table Entry Object Size	13.2.2.20.53
0x00522038	GLHMC_PEARPMAX	Private Memory Protocol Engine ARP Table Entry Max	13.2.2.20.54
0x0052203C	GLHMC_PEMROBJSZ	Private Memory PE Memory Region Table Entry Object Size	13.2.2.20.55
0x00522040	GLHMC_PEMRMAX	Private Memory Protocol Engine Memory Registration Max	13.2.2.20.56
0x00522044	GLHMC_PEXFOBJSZ	Private Memory PE Xmit FIFO Object Size	13.2.2.20.57
0x00522048	GLHMC_PEXFMAX	Private Memory Protocol Engine Transmit FIFO Entry Max	13.2.2.20.58
0x0052204C	GLHMC_PEXFFLMAX	Private Memory Protocol Engine Transmit FIFO Free List Max	13.2.2.20.59
0x00522050	GLHMC_PEQ1OBJSZ	Private Memory PE IRRQ Object Size	13.2.2.20.60
0x00522054	GLHMC_PEQ1MAX	Private Memory Protocol Engine Q1 Max	13.2.2.20.61
0x00522058	GLHMC_PEQ1FLMAX	Private Memory Protocol Engine Q1 Free List Max	13.2.2.20.62
0x0052205C	GLHMC_FSIMCOBJSZ	Private Memory FSI Multicast Group Object Size	13.2.2.20.63
0x00522060	GLHMC_FSIMCMAX	Private Memory FSI Multicast Group Max	13.2.2.20.64
0x00522064	GLHMC_FSIAVOBJSZ	Private Memory FSI Address Vector Object Size	13.2.2.20.65

Table 13-28. PF - Host Memory Cache Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00522068	GLHMC_FSIAVMAX	Private Memory FSI Address Vector Max	13.2.2.20.66
0x0052206C	GLHMC_PEPBLMAX	Private Memory Protocol Engine Physical Buffer List Max	13.2.2.20.67
0x00522074	GLHMC_FWSDDATALOW	Private Memory Space Segment Descriptor Data Low	13.2.2.20.68
0x00522078	GLHMC_FWSDDATAHIGH	Private Memory Space Segment Descriptor Data High	13.2.2.20.69
0x0052207C	GLHMC_FWPDINV	Private Memory Space Page Descriptor Invalidate	13.2.2.20.70
0x00522080	GLHMC_PETIMEROBJSZ	Private Memory PE Timer Object Size	13.2.2.20.71
0x00522084	GLHMC_PETIMERMAX	Private Memory PE Timer Object Max	13.2.2.20.72
0x00522098	GLHMC_PERRFOBJSZ	Private Memory Protocol Engine Read Response Entry Object Size	13.2.2.20.73
0x0052209C	GLHMC_PERRFMAX	Private Memory Protocol Engine Read Response FIFO Entry Max	13.2.2.20.74
0x005220A0	GLHMC_PERRFFLMAX	Private Memory Protocol Engine Read Response FIFO Free List Max	13.2.2.20.75
0x005220A4	GLHMC_PEOOISCFFLMAX	Private Memory Protocol Engine Out of Order Send Completion (OOISC) FIFO Free List Max	13.2.2.20.76
0x005220EC	GLHMC_DBQPMAX	Private Memory Protocol Engine Queue Pair Max	13.2.2.20.77
0x005220F0	GLHMC_DBCQMAX	Private Memory Protocol Engine Completion Queue Max	13.2.2.20.78
0x00524000 + 0x4*n, n=0...7	GLHMC_PEQPBASE[n]	FPM PE QP Base	13.2.2.20.79
0x00524100 + 0x4*n, n=0...7	GLHMC_PEQPCNT[n]	FPM PE QP Object Count	13.2.2.20.80
0x00524200 + 0x4*n, n=0...7	GLHMC_PECQBASE[n]	FPM PE CQ Base	13.2.2.20.81
0x00524300 + 0x4*n, n=0...7	GLHMC_PECQCNT[n]	FPM PE CQ Object Count	13.2.2.20.82
0x00524600 + 0x4*n, n=0...7	GLHMC_PEHTEBASE[n]	FPM PE Hash Table Entry Base	13.2.2.20.83
0x00524700 + 0x4*n, n=0...7	GLHMC_PEHTCNT[n]	FPM PE Hash Table Object Count	13.2.2.20.84
0x00524800 + 0x4*n, n=0...7	GLHMC_PEARPBASE[n]	FPM PE ARP Table Base	13.2.2.20.85
0x00524900 + 0x4*n, n=0...7	GLHMC_PEARPCNT[n]	FPM PE ARP Table Object Count	13.2.2.20.86
0x00524A00 + 0x4*n, n=0...7	GLHMC_APBVTINUSEBASE[n]	FPM PE APBVT In-Use Base	13.2.2.20.87
0x00524C00 + 0x4*n, n=0...7	GLHMC_PEMRBASE[n]	FPM PE MRT Base	13.2.2.20.88
0x00524D00 + 0x4*n, n=0...7	GLHMC_PEMRCNT[n]	FPM PE Memory Region Table Object Count	13.2.2.20.89
0x00524E00 + 0x4*n, n=0...7	GLHMC_PEXFBASE[n]	FPM PE Xmit FIFO Base	13.2.2.20.90
0x00524F00 + 0x4*n, n=0...7	GLHMC_PEXFCNT[n]	FPM PE Xmit FIFO Object Count	13.2.2.20.91

Table 13-28. PF - Host Memory Cache Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00525000 + 0x4*n, n=0...7	GLHMC_PEXFFLBASE[n]	FPM PE Xmit FIFO Free List Base	13.2.2.20.92
0x00525200 + 0x4*n, n=0...7	GLHMC_PEQ1BASE[n]	FPM PE IRRQ Base	13.2.2.20.93
0x00525300 + 0x4*n, n=0...7	GLHMC_PEQ1CNT[n]	FPM PE IRRQ Object Count	13.2.2.20.94
0x00525400 + 0x4*n, n=0...7	GLHMC_PEQ1FLBASE[n]	FPM PE IRRQ Free List Base	13.2.2.20.95
0x00525600 + 0x4*n, n=0...7	GLHMC_FSIAVBASE[n]	FPM FSI Address Vector Base	13.2.2.20.96
0x00525700 + 0x4*n, n=0...7	GLHMC_FSIAVCNT[n]	FPM FSI Address Vector Object Count	13.2.2.20.97
0x00525800 + 0x4*n, n=0...7	GLHMC_PEPBLBASE[n]	FPM PE Physical Buffer List Base	13.2.2.20.98
0x00525900 + 0x4*n, n=0...7	GLHMC_PEPBLCNT[n]	FPM PE PBL Object Count	13.2.2.20.99
0x00525A00 + 0x4*n, n=0...7	GLHMC_PETIMERBASE[n]	FPM PE Timer Base	13.2.2.20.100
0x00525B00 + 0x4*n, n=0...7	GLHMC_PETIMERCNT[n]	FPM PE Timer Object Count	13.2.2.20.101
0x00526000 + 0x4*n, n=0...7	GLHMC_FSIMCBASE[n]	FPM FSI Multicast Group Base	13.2.2.20.102
0x00526100 + 0x4*n, n=0...7	GLHMC_FSIMCCNT[n]	FPM FSI Multicast Group Object Count	13.2.2.20.103
0x00526200 + 0x4*n, n=0...7	GLHMC_PEHDRBASE[n]	FPM PE Header Base	13.2.2.20.104
0x00526300 + 0x4*n, n=0...7	GLHMC_PEHDRCNT[n]	FPM PE Header Object Count	13.2.2.20.105
0x00526400 + 0x4*n, n=0...7	GLHMC_PEMDBASE[n]	FPM PE Metadata Base	13.2.2.20.106
0x00526500 + 0x4*n, n=0...7	GLHMC_PEMDCNT[n]	FPM PE Metadata Object Count	13.2.2.20.107
0x00526600 + 0x4*n, n=0...7	GLHMC_PEOOISCBASE[n]	FPM PE Out of Order Send Completion Base	13.2.2.20.108
0x00526700 + 0x4*n, n=0...7	GLHMC_PEOOISCCNT[n]	FPM PE Out of Order Send Completion Object Count	13.2.2.20.109
0x00526800 + 0x4*n, n=0...7	GLHMC_PERRFBASE[n]	FPM PE Read Response Base	13.2.2.20.110
0x00526900 + 0x4*n, n=0...7	GLHMC_PERRFCNT[n]	FPM PE Read Response Object Count	13.2.2.20.111
0x00526A00 + 0x4*n, n=0...7	GLHMC_PERRFFLBASE[n]	FPM PE Read Response FIFO Free List Base	13.2.2.20.112
0x00526B00 + 0x4*n, n=0...7	GLHMC_PERRFFLCNT_PMAT[n]	FPM PE Read Response FIFO Free List Object Count	13.2.2.20.113
0x00526C00 + 0x4*n, n=0...7	GLHMC_PEOOISCFFLBASE[n]	FPM PE Out of Order Send Completion (OOISC) FIFO Free List Base	13.2.2.20.114
0x00526D00 + 0x4*n, n=0...7	GLHMC_PEOOISCFFLCNT_PMAT[n]	FPM PE Out of Order Send Completion (OOISC) FIFO Free List Object Count	13.2.2.20.115

Table 13-28. PF - Host Memory Cache Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00528100 + 0x4*n, n=0...31	GLHMC_VFSDDATALOW[n]	Private Memory Space VF Segment Descriptor Data Low	13.2.2.20.116
0x00528200 + 0x4*n, n=0...31	GLHMC_VFSDDATAHIGH[n]	Private Memory Space VF Segment Descriptor Data High	13.2.2.20.117
0x00528300 + 0x4*n, n=0...31	GLHMC_VFPDINV[n]	Private Memory Space Page Descriptor Invalidate	13.2.2.20.118
0x00528800 + 0x4*n, n=0...31	GLHMC_VFSDPART[n]	Private Memory Segment Table Partitioning Registers	13.2.2.20.119
0x0052C000 + 0x4*n, n=0...31	GLHMC_VFPEQPBASE[n]	FPM PE QP Base	13.2.2.20.120
0x0052C100 + 0x4*n, n=0...31	GLHMC_VFPEQPCNT[n]	FPM PE QP Object Count	13.2.2.20.121
0x0052C200 + 0x4*n, n=0...31	GLHMC_VFPECQBASE[n]	FPM PE CQ Base	13.2.2.20.122
0x0052C300 + 0x4*n, n=0...31	GLHMC_VFPECQCNT[n]	FPM PE CQ Object Count	13.2.2.20.123
0x0052C600 + 0x4*n, n=0...31	GLHMC_VFPEHTEBASE[n]	FPM PE Hash Table Entry Base	13.2.2.20.124
0x0052C700 + 0x4*n, n=0...31	GLHMC_VFPEHTCNT[n]	FPM PE Hash Table Object Count	13.2.2.20.125
0x0052C800 + 0x4*n, n=0...31	GLHMC_VFPEARPBASE[n]	FPM PE ARP Table Base	13.2.2.20.126
0x0052C900 + 0x4*n, n=0...31	GLHMC_VFPEARPCNT[n]	FPM PE ARP Table Object Count	13.2.2.20.127
0x0052CA00 + 0x4*n, n=0...31	GLHMC_VFAPBVTINUSEBASE[n]	FPM PE APBVT In-Use Base	13.2.2.20.128
0x0052CC00 + 0x4*n, n=0...31	GLHMC_VFPEMRBASE[n]	FPM PE MRT Base	13.2.2.20.129
0x0052CD00 + 0x4*n, n=0...31	GLHMC_VFPEMRCNT[n]	FPM PE Memory Region Table Object Count	13.2.2.20.130
0x0052CE00 + 0x4*n, n=0...31	GLHMC_VFPEXFBASE[n]	FPM PE Xmit FIFO Base	13.2.2.20.131
0x0052CF00 + 0x4*n, n=0...31	GLHMC_VFPEXFCNT[n]	FPM PE Xmit FIFO Object Count	13.2.2.20.132
0x0052D000 + 0x4*n, n=0...31	GLHMC_VFPEXFFLBASE[n]	FPM PE Xmit FIFO Free List Base	13.2.2.20.133
0x0052D200 + 0x4*n, n=0...31	GLHMC_VFPEQ1BASE[n]	FPM PE IRRQ Base	13.2.2.20.134
0x0052D300 + 0x4*n, n=0...31	GLHMC_VFPEQ1CNT[n]	FPM PE IRRQ Object Count	13.2.2.20.135
0x0052D400 + 0x4*n, n=0...31	GLHMC_VFPEQ1FLBASE[n]	FPM PE IRRQ Free List Base	13.2.2.20.136
0x0052D600 + 0x4*n, n=0...31	GLHMC_VFFSIAVBASE[n]	FPM FSI Address Vector Base	13.2.2.20.137
0x0052D700 + 0x4*n, n=0...31	GLHMC_VFFSIAVCNT[n]	FPM FSI Address Vector Object Count	13.2.2.20.138
0x0052D800 + 0x4*n, n=0...31	GLHMC_VFPEPBLBASE[n]	FPM PE Physical Buffer List Base	13.2.2.20.139

Table 13-28. PF - Host Memory Cache Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0052D900 + 0x4*n, n=0...31	GLHMC_VFPEPBLCNT[n]	FPM PE PBL Object Count	13.2.2.20.140
0x0052DA00 + 0x4*n, n=0...31	GLHMC_VFPETIMERBASE[n]	FPM PE Timer Base	13.2.2.20.141
0x0052DB00 + 0x4*n, n=0...31	GLHMC_VFPETIMERCNT[n]	FPM PE Timer Object Count	13.2.2.20.142
0x0052E000 + 0x4*n, n=0...31	GLHMC_VFFSIMCBASE[n]	FPM FSI Multicast Group Base	13.2.2.20.143
0x0052E100 + 0x4*n, n=0...31	GLHMC_VFFSIMCCNT[n]	FPM FSI Multicast Group Object Count	13.2.2.20.144
0x0052E200 + 0x4*n, n=0...31	GLHMC_VFPEHDRBASE[n]	FPM PE Header Base	13.2.2.20.145
0x0052E300 + 0x4*n, n=0...31	GLHMC_VFPEHDRCNT[n]	FPM PE Header Object Count	13.2.2.20.146
0x0052E400 + 0x4*n, n=0...31	GLHMC_VFPEMDBASE[n]	FPM PE Metadata Base	13.2.2.20.147
0x0052E500 + 0x4*n, n=0...31	GLHMC_VFPEMDCNT[n]	FPM PE Metadata Object Count	13.2.2.20.148
0x0052E600 + 0x4*n, n=0...31	GLHMC_VFPEOOISCBASE[n]	FPM PE Out of Order Send Completion Base	13.2.2.20.149
0x0052E700 + 0x4*n, n=0...31	GLHMC_VFPEOOISCCNT[n]	FPM PE Out of Order Send Completion Object Count	13.2.2.20.150
0x0052E800 + 0x4*n, n=0...31	GLHMC_VFPERRFBASE[n]	FPM PE Read Response Base	13.2.2.20.151
0x0052E900 + 0x4*n, n=0...31	GLHMC_VFPERRFCNT[n]	FPM PE Read Response Object Count	13.2.2.20.152
0x0052EA00 + 0x4*n, n=0...31	GLHMC_VFPERRFFLBASE[n]	FPM PE Read Response FIFO Free List Base	13.2.2.20.153
0x0052EC00 + 0x4*n, n=0...31	GLHMC_VFPEOOISCFFLBASE[n]	FPM PE Out of Order Send Completion (OOISC) FIFO Free List Base	13.2.2.20.154
0x00530048	GLPDOC_CACHESIZE	PDOC Cache Attributes	13.2.2.20.155

Table 13-29. PF - Context Manager Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x005046B4	GLCM_PE_CACHESIZE	CMPE Cache Attributes	13.2.2.21.1

Table 13-30. PF - Control Queues Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00080000	PF_FW_ATQBAL	PF Firmware Admin Transmit Queue Base Address Low	13.2.2.22.1
0x00080040	GL_FW_TOOL_ATQBAL	Global Tools Firmware Admin Transmit Queue Base Address Low	13.2.2.22.2
0x00080044	PF0_FW_PSM_ATQBAL	PF0 PSM Firmware Admin Transmit Queue Base Address Low	13.2.2.22.3

Table 13-30. PF - Control Queues Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00080048	PF0_FW_HLP_ATQBAL	PF0 HLP Firmware Admin Transmit Queue Base Address Low	13.2.2.22.4
0x00080080	PF_FW_ARQBAL	PF Firmware Admin Receive Queue Base Address Low	13.2.2.22.5
0x000800C0	GL_FW_TOOL_ARQBAL	Global Tools Firmware Admin Receive Queue Base Address Low	13.2.2.22.6
0x000800C4	PF0_FW_PSM_ARQBAL	PF0 PSM Firmware Admin Receive Queue Base Address Low	13.2.2.22.7
0x000800C8	PF0_FW_HLP_ARQBAL	PF0 HLP Firmware Admin Receive Queue Base Address Low	13.2.2.22.8
0x00080100	PF_FW_ATQBAH	PF Firmware Admin Transmit Queue Base Address High	13.2.2.22.9
0x00080140	GL_FW_TOOL_ATQBAH	Global Tools Firmware Admin Transmit Queue Base Address High	13.2.2.22.10
0x00080144	PF0_FW_PSM_ATQBAH	PF0 PSM Firmware Admin Transmit Queue Base Address High	13.2.2.22.11
0x00080148	PF0_FW_HLP_ATQBAH	PF0 HLP Firmware Admin Transmit Queue Base Address High	13.2.2.22.12
0x00080180	PF_FW_ARQBAH	PF Firmware Admin Receive Queue Base Address High	13.2.2.22.13
0x000801C0	GL_FW_TOOL_ARQBAH	Global Tools Firmware Admin Receive Queue Base Address High	13.2.2.22.14
0x000801C4	PF0_FW_PSM_ARQBAH	PF0 PSM Firmware Admin Receive Queue Base Address High	13.2.2.22.15
0x000801C8	PF0_FW_HLP_ARQBAH	PF0 HLP Firmware Admin Receive Queue Base Address High	13.2.2.22.16
0x00080200	PF_FW_ATQLEN	PF Firmware Admin Transmit Queue Length	13.2.2.22.17
0x00080240	GL_FW_TOOL_ATQLEN	Global Tools Firmware Admin Transmit Queue Length	13.2.2.22.18
0x00080244	PF0_FW_PSM_ATQLEN	PF0 PSM Firmware Admin Transmit Queue Length	13.2.2.22.19
0x00080248	PF0_FW_HLP_ATQLEN	PF0 HLP Firmware Admin Transmit Queue Length	13.2.2.22.20
0x00080280	PF_FW_ARQLEN	PF Firmware Admin Receive Queue Length	13.2.2.22.21
0x000802C0	GL_FW_TOOL_ARQLEN	Global Tools Firmware Admin Receive Queue Length	13.2.2.22.22
0x000802C4	PF0_FW_PSM_ARQLEN	PF0 PSM Firmware Admin Receive Queue Length	13.2.2.22.23
0x000802C8	PF0_FW_HLP_ARQLEN	PF0 HLP Firmware Admin Receive Queue Length	13.2.2.22.24
0x00080300	PF_FW_ATQH	PF Firmware Admin Transmit Head	13.2.2.22.25
0x00080340	GL_FW_TOOL_ATQH	Global Tools Firmware Admin Transmit Head	13.2.2.22.26
0x00080344	PF0_FW_PSM_ATQH	PF0 PSM Firmware Admin Transmit Head	13.2.2.22.27
0x00080348	PF0_FW_HLP_ATQH	PF0 HLP Firmware Admin Transmit Head	13.2.2.22.28
0x00080380	PF_FW_ARQH	PF Firmware Admin Receive Queue Head	13.2.2.22.29

Table 13-30. PF - Control Queues Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x000803C0	GL_FW_TOOL_ARQH	Global Tools Firmware Admin Receive Queue Head	13.2.2.22.30
0x000803C4	PF0_FW_PSM_ARQH	PF0 PSM Firmware Admin Receive Queue Head	13.2.2.22.31
0x000803C8	PF0_FW_HLP_ARQH	PF0 HLP Firmware Admin Receive Queue Head	13.2.2.22.32
0x00080400	PF_FW_ATQT	PF Firmware Admin Transmit Tail	13.2.2.22.33
0x00080440	GL_FW_TOOL_ATQT	Global Tools Firmware Admin Transmit Tail	13.2.2.22.34
0x00080444	PF0_FW_PSM_ATQT	PF0 PSM Firmware Admin Transmit Tail	13.2.2.22.35
0x00080448	PF0_FW_HLP_ATQT	PF0 HLP Firmware Admin Transmit Tail	13.2.2.22.36
0x00080480	PF_FW_ARQT	PF Firmware Admin Receive Queue Tail	13.2.2.22.37
0x000804C0	GL_FW_TOOL_ARQT	Global Tools Firmware Admin Receive Queue Tail	13.2.2.22.38
0x000804C4	PF0_FW_PSM_ARQT	PF0 PSM Firmware Admin Receive Queue Tail	13.2.2.22.39
0x000804C8	PF0_FW_HLP_ARQT	PF0 HLP Firmware Admin Receive Queue Tail	13.2.2.22.40
0x00081000 + 0x4*n, n=0...1023	GL_HIBA[n]	Tools Mailbox HOST Interface Buffer Area	13.2.2.22.41
0x00082000 + 0x4*n, n=0...15	GL_HIDA[n]	Tools Mailbox HOST Interface Descriptor Area	13.2.2.22.42
0x00082040	GL_HICR	Tools Mailbox HOST Interface Control Register	13.2.2.22.43
0x00082044	GL_HICR_EN	Tools Mailbox HOST Interface Enable Register	13.2.2.22.44
0x00220000 + 0x4*VSI, VSI=0...767	VSI_MBX_ATQBAL[VSI]	VSI Mailbox Transmit Queue Base Address Low	13.2.2.22.45
0x00221000 + 0x4*VSI, VSI=0...767	VSI_MBX_ATQBAH[VSI]	VSI Mailbox Transmit Queue Base Address High	13.2.2.22.46
0x00222000 + 0x4*VSI, VSI=0...767	VSI_MBX_ATQLEN[VSI]	VSI Mailbox Transmit Queue Length	13.2.2.22.47
0x00223000 + 0x4*VSI, VSI=0...767	VSI_MBX_ATQH[VSI]	VSI Mailbox Transmit Head	13.2.2.22.48
0x00224000 + 0x4*VSI, VSI=0...767	VSI_MBX_ATQT[VSI]	VSI Mailbox Transmit Tail	13.2.2.22.49
0x00225000 + 0x4*VSI, VSI=0...767	VSI_MBX_ARQBAL[VSI]	VSI Mailbox Receive Queue Base Address Low	13.2.2.22.50
0x00226000 + 0x4*VSI, VSI=0...767	VSI_MBX_ARQBAH[VSI]	VSI Mailbox Receive Queue Base Address High	13.2.2.22.51
0x00227000 + 0x4*VSI, VSI=0...767	VSI_MBX_ARQLEN[VSI]	VSI Mailbox Receive Queue Length	13.2.2.22.52
0x00228000 + 0x4*VSI, VSI=0...767	VSI_MBX_ARQH[VSI]	VSI Mailbox Receive Head	13.2.2.22.53
0x00229000 + 0x4*VSI, VSI=0...767	VSI_MBX_ARQT[VSI]	VSI Mailbox Receive Tail	13.2.2.22.54
0x0022A000 + 0x4*VF, VF=0...255	VF_MBX_ATQBAL[VF]	VF Mailbox Transmit Queue Base Address Low	13.2.2.22.55

Table 13-30. PF - Control Queues Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0022A400 + 0x4*VF, VF=0...255	VF_MBX_ATQBAH[VF]	VF Mailbox Transmit Queue Base Address High	13.2.2.22.56
0x0022A800 + 0x4*VF, VF=0...255	VF_MBX_ATQLEN[VF]	VF Mailbox Transmit Queue Length	13.2.2.22.57
0x0022AC00 + 0x4*VF, VF=0...255	VF_MBX_ATQH[VF]	VF Mailbox Transmit Head	13.2.2.22.58
0x0022B000 + 0x4*VF, VF=0...255	VF_MBX_ATQT[VF]	VF Mailbox Transmit Tail	13.2.2.22.59
0x0022B400 + 0x4*VF, VF=0...255	VF_MBX_ARQBAL[VF]	VF Mailbox Receive Queue Base Address Low	13.2.2.22.60
0x0022B800 + 0x4*VF, VF=0...255	VF_MBX_ARQBAH[VF]	VF Mailbox Receive Queue Base Address High	13.2.2.22.61
0x0022BC00 + 0x4*VF, VF=0...255	VF_MBX_ARQLEN[VF]	VF Mailbox Receive Queue Length	13.2.2.22.62
0x0022C000 + 0x4*VF, VF=0...255	VF_MBX_ARQH[VF]	VF Mailbox Receive Head	13.2.2.22.63
0x0022C400 + 0x4*VF, VF=0...255	VF_MBX_ARQT[VF]	VF Mailbox Receive Tail	13.2.2.22.64
0x0022C800 + 0x4*VF128, VF128=0...127	VF_MBX_CPM_ATQBAL[VF128]	VF CPM Mailbox Transmit Queue Base Address Low	13.2.2.22.65
0x0022CA00 + 0x4*VF128, VF128=0...127	VF_MBX_CPM_ATQBAH[VF128]	VF CPM Mailbox Transmit Queue Base Address High	13.2.2.22.66
0x0022CC00 + 0x4*VF128, VF128=0...127	VF_MBX_CPM_ATQLEN[VF128]	VF CPM Mailbox Transmit Queue Length	13.2.2.22.67
0x0022CE00 + 0x4*VF128, VF128=0...127	VF_MBX_CPM_ATQH[VF128]	VF CPM Mailbox Transmit Head	13.2.2.22.68
0x0022D000 + 0x4*VF128, VF128=0...127	VF_MBX_CPM_ATQT[VF128]	VF CPM Mailbox Transmit Tail	13.2.2.22.69
0x0022D200 + 0x4*VF128, VF128=0...127	VF_MBX_CPM_ARQBAL[VF128]	VF CPM Mailbox Receive Queue Base Address Low	13.2.2.22.70
0x0022D400 + 0x4*VF128, VF128=0...127	VF_MBX_CPM_ARQBAH[VF128]	VF CPM Mailbox Receive Queue Base Address High	13.2.2.22.71
0x0022D600 + 0x4*VF128, VF128=0...127	VF_MBX_CPM_ARQLEN[VF128]	VF CPM Mailbox Receive Queue Length	13.2.2.22.72
0x0022D800 + 0x4*VF128, VF128=0...127	VF_MBX_CPM_ARQH[VF128]	VF CPM Mailbox Receive Head	13.2.2.22.73
0x0022DA00 + 0x4*VF128, VF128=0...127	VF_MBX_CPM_ARQT[VF128]	VF CPM Mailbox Receive Tail	13.2.2.22.74
0x0022DC00 + 0x4*VF16, VF16=0...15	VF_MBX_HLP_ATQBAL[VF16]	VF HLP Mailbox Transmit Queue Base Address Low	13.2.2.22.75
0x0022DC40 + 0x4*VF16, VF16=0...15	VF_MBX_HLP_ATQBAH[VF16]	VF HLP Mailbox Transmit Queue Base Address High	13.2.2.22.76
0x0022DC80 + 0x4*VF16, VF16=0...15	VF_MBX_HLP_ATQLEN[VF16]	VF HLP Mailbox Transmit Queue Length	13.2.2.22.77
0x0022DCC0 + 0x4*VF16, VF16=0...15	VF_MBX_HLP_ATQH[VF16]	VF HLP Mailbox Transmit Head	13.2.2.22.78
0x0022DD00 + 0x4*VF16, VF16=0...15	VF_MBX_HLP_ATQT[VF16]	VF HLP Mailbox Transmit Tail	13.2.2.22.79

Table 13-30. PF - Control Queues Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0022DD40 + 0x4*VF16, VF16=0...15	VF_MBX_HLP_ARQBAL[VF16]	VF HLP Mailbox Receive Queue Base Address Low	13.2.2.22.80
0x0022DD80 + 0x4*VF16, VF16=0...15	VF_MBX_HLP_ARQBAH[VF16]	VF HLP Mailbox Receive Queue Base Address High	13.2.2.22.81
0x0022DDC0 + 0x4*VF16, VF16=0...15	VF_MBX_HLP_ARQLEN[VF16]	VF HLP Mailbox Receive Queue Length	13.2.2.22.82
0x0022DE00 + 0x4*VF16, VF16=0...15	VF_MBX_HLP_ARQH[VF16]	VF HLP Mailbox Receive Head	13.2.2.22.83
0x0022DE40 + 0x4*VF16, VF16=0...15	VF_MBX_HLP_ARQT[VF16]	VF HLP Mailbox Receive Tail	13.2.2.22.84
0x0022DE80 + 0x4*VF16, VF16=0...15	VF_MBX_PSM_ATQBAL[VF16]	VF PSM Mailbox Transmit Queue Base Address Low	13.2.2.22.85
0x0022DEC0 + 0x4*VF16, VF16=0...15	VF_MBX_PSM_ATQBAH[VF16]	VF PSM Mailbox Transmit Queue Base Address High	13.2.2.22.86
0x0022DF00 + 0x4*VF16, VF16=0...15	VF_MBX_PSM_ATQLEN[VF16]	VF PSM Mailbox Transmit Queue Length	13.2.2.22.87
0x0022DF40 + 0x4*VF16, VF16=0...15	VF_MBX_PSM_ATQH[VF16]	VF PSM Mailbox Transmit Head	13.2.2.22.88
0x0022DF80 + 0x4*VF16, VF16=0...15	VF_MBX_PSM_ATQT[VF16]	VF PSM Mailbox Transmit Tail	13.2.2.22.89
0x0022DFC0 + 0x4*VF16, VF16=0...15	VF_MBX_PSM_ARQBAL[VF16]	VF PSM Mailbox Receive Queue Base Address Low	13.2.2.22.90
0x0022E000 + 0x4*VF16, VF16=0...15	VF_MBX_PSM_ARQBAH[VF16]	VF PSM Mailbox Receive Queue Base Address High	13.2.2.22.91
0x0022E040 + 0x4*VF16, VF16=0...15	VF_MBX_PSM_ARQLEN[VF16]	VF PSM Mailbox Receive Queue Length	13.2.2.22.92
0x0022E080 + 0x4*VF16, VF16=0...15	VF_MBX_PSM_ARQH[VF16]	VF PSM Mailbox Receive Head	13.2.2.22.93
0x0022E0C0 + 0x4*VF16, VF16=0...15	VF_MBX_PSM_ARQT[VF16]	VF PSM Mailbox Receive Tail	13.2.2.22.94
0x0022E100	PF_MBX_ATQBAL	PF Mailbox Transmit Queue Base Address Low	13.2.2.22.95
0x0022E180	PF_MBX_ATQBAH	PF Mailbox Transmit Queue Base Address High	13.2.2.22.96
0x0022E200	PF_MBX_ATQLEN	PF Mailbox Transmit Queue Length	13.2.2.22.97
0x0022E280	PF_MBX_ATQH	PF Mailbox Transmit Head	13.2.2.22.98
0x0022E300	PF_MBX_ATQT	PF Mailbox Transmit Tail	13.2.2.22.99
0x0022E380	PF_MBX_ARQBAL	PF Mailbox Receive Queue Base Address Low	13.2.2.22.100
0x0022E400	PF_MBX_ARQBAH	PF Mailbox Receive Queue Base Address High	13.2.2.22.101
0x0022E480	PF_MBX_ARQLEN	PF Mailbox Receive Queue Length	13.2.2.22.102
0x0022E500	PF_MBX_ARQH	PF Mailbox Receive Head	13.2.2.22.103
0x0022E580	PF_MBX_ARQT	PF Mailbox Receive Tail	13.2.2.22.104
0x0022E5C0	PF0_MBX_CPM_ATQBAL	PF0 CPM Mailbox Transmit Queue Base Address Low	13.2.2.22.105

Table 13-30. PF - Control Queues Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0022E5C4	PF0_MBX_CPM_ATQBAH	PF0 CPM Mailbox Transmit Queue Base Address High	13.2.2.22.106
0x0022E5C8	PF0_MBX_CPM_ATQLEN	PF0 CPM Mailbox Transmit Queue Length	13.2.2.22.107
0x0022E5CC	PF0_MBX_CPM_ATQH	PF0 CPM Mailbox Transmit Head	13.2.2.22.108
0x0022E5D0	PF0_MBX_CPM_ATQT	PF0 CPM Mailbox Transmit Tail	13.2.2.22.109
0x0022E5D4	PF0_MBX_CPM_ARQBAL	PF0 CPM Mailbox Receive Queue Base Address Low	13.2.2.22.110
0x0022E5D8	PF0_MBX_CPM_ARQBAH	PF0 CPM Mailbox Receive Queue Base Address High	13.2.2.22.111
0x0022E5DC	PF0_MBX_CPM_ARQLEN	PF0 CPM Mailbox Receive Queue Length	13.2.2.22.112
0x0022E5E0	PF0_MBX_CPM_ARQH	PF0 CPM Mailbox Receive Head	13.2.2.22.113
0x0022E5E4	PF0_MBX_CPM_ARQT	PF0 CPM Mailbox Receive Tail	13.2.2.22.114
0x0022E5E8	PF0_MBX_HLP_ATQBAL	PF0 HLP Mailbox Transmit Queue Base Address Low	13.2.2.22.115
0x0022E5EC	PF0_MBX_HLP_ATQBAH	PF0 HLP Mailbox Transmit Queue Base Address High	13.2.2.22.116
0x0022E5F0	PF0_MBX_HLP_ATQLEN	PF0 HLP Mailbox Transmit Queue Length	13.2.2.22.117
0x0022E5F4	PF0_MBX_HLP_ATQH	PF0 HLP Mailbox Transmit Head	13.2.2.22.118
0x0022E5F8	PF0_MBX_HLP_ATQT	PF0 HLP Mailbox Transmit Tail	13.2.2.22.119
0x0022E5FC	PF0_MBX_HLP_ARQBAL	PF0 HLP Mailbox Receive Queue Base Address Low	13.2.2.22.120
0x0022E600	PF0_MBX_HLP_ARQBAH	PF0 HLP Mailbox Receive Queue Base Address High	13.2.2.22.121
0x0022E604	PF0_MBX_HLP_ARQLEN	PF0 HLP Mailbox Receive Queue Length	13.2.2.22.122
0x0022E608	PF0_MBX_HLP_ARQH	PF0 HLP Mailbox Receive Head	13.2.2.22.123
0x0022E60C	PF0_MBX_HLP_ARQT	PF0 HLP Mailbox Receive Tail	13.2.2.22.124
0x0022E610	PF0_MBX_PSM_ATQBAL	PF0 PSM Mailbox Transmit Queue Base Address Low	13.2.2.22.125
0x0022E614	PF0_MBX_PSM_ATQBAH	PF0 PSM Mailbox Transmit Queue Base Address High	13.2.2.22.126
0x0022E618	PF0_MBX_PSM_ATQLEN	PF0 PSM Mailbox Transmit Queue Length	13.2.2.22.127
0x0022E61C	PF0_MBX_PSM_ATQH	PF0 PSM Mailbox Transmit Head	13.2.2.22.128
0x0022E620	PF0_MBX_PSM_ATQT	PF0 PSM Mailbox Transmit Tail	13.2.2.22.129
0x0022E624	PF0_MBX_PSM_ARQBAL	PF0 PSM Mailbox Receive Queue Base Address Low	13.2.2.22.130
0x0022E628	PF0_MBX_PSM_ARQBAH	PF0 PSM Mailbox Receive Queue Base Address High	13.2.2.22.131
0x0022E62C	PF0_MBX_PSM_ARQLEN	PF0 PSM Mailbox Receive Queue Length	13.2.2.22.132
0x0022E630	PF0_MBX_PSM_ARQH	PF0 PSM Mailbox Receive Head	13.2.2.22.133
0x0022E634	PF0_MBX_PSM_ARQT	PF0 PSM Mailbox Receive Tail	13.2.2.22.134
0x0022E638	PF0_SB_CPM_ATQBAL	PF0 CPM Sideband Transmit Queue Base Address Low	13.2.2.22.135

Table 13-30. PF - Control Queues Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0022E63C	PF0_SB_CPM_ATQBAH	PF0 CPM Sideband Transmit Queue Base Address High	13.2.2.22.136
0x0022E640	PF0_SB_CPM_ATQLEN	PF0 CPM Sideband Transmit Queue Length	13.2.2.22.137
0x0022E644	PF0_SB_CPM_ATQH	PF0 CPM Sideband Transmit Head	13.2.2.22.138
0x0022E648	PF0_SB_CPM_ATQT	PF0 CPM Sideband Transmit Tail	13.2.2.22.139
0x0022E64C	PF0_SB_CPM_ARQBAL	PF0 CPM Sideband Receive Queue Base Address Low	13.2.2.22.140
0x0022E650	PF0_SB_CPM_ARQBAH	PF0 CPM Sideband Receive Queue Base Address High	13.2.2.22.141
0x0022E654	PF0_SB_CPM_ARQLEN	PF0 CPM Sideband Receive Queue Length	13.2.2.22.142
0x0022E658	PF0_SB_CPM_ARQH	PF0 CPM Sideband Receive Head	13.2.2.22.143
0x0022E65C	PF0_SB_CPM_ARQT	PF0 CPM Sideband Receive Tail	13.2.2.22.144
0x0022E800 + 0x4*VF128, VF128=0...127	VF_SB_CPM_ATQBAL[VF128]	VF CPM Sideband Transmit Queue Base Address Low	13.2.2.22.145
0x0022EA00 + 0x4*VF128, VF128=0...127	VF_SB_CPM_ATQBAH[VF128]	VF CPM Sideband Transmit Queue Base Address High	13.2.2.22.146
0x0022EC00 + 0x4*VF128, VF128=0...127	VF_SB_CPM_ATQLEN[VF128]	VF CPM Sideband Transmit Queue Length	13.2.2.22.147
0x0022EE00 + 0x4*VF128, VF128=0...127	VF_SB_CPM_ATQH[VF128]	VF CPM Sideband Transmit Head	13.2.2.22.148
0x0022F000 + 0x4*VF128, VF128=0...127	VF_SB_CPM_ATQT[VF128]	VF CPM Sideband Transmit Tail	13.2.2.22.149
0x0022F200 + 0x4*VF128, VF128=0...127	VF_SB_CPM_ARQBAL[VF128]	VF CPM Sideband Receive Queue Base Address Low	13.2.2.22.150
0x0022F400 + 0x4*VF128, VF128=0...127	VF_SB_CPM_ARQBAH[VF128]	VF CPM Sideband Receive Queue Base Address High	13.2.2.22.151
0x0022F600 + 0x4*VF128, VF128=0...127	VF_SB_CPM_ARQLEN[VF128]	VF CPM Sideband Receive Queue Length	13.2.2.22.152
0x0022F800 + 0x4*VF128, VF128=0...127	VF_SB_CPM_ARQH[VF128]	VF CPM Sideband Receive Head	13.2.2.22.153
0x0022FA00 + 0x4*VF128, VF128=0...127	VF_SB_CPM_ARQT[VF128]	VF CPM Sideband Receive Tail	13.2.2.22.154
0x0022FC00	PF_SB_ATQBAL	PF Sideband Transmit Queue Base Address Low	13.2.2.22.155
0x0022FC80	PF_SB_ATQBAH	PF Sideband Transmit Queue Base Address High	13.2.2.22.156
0x0022FD00	PF_SB_ATQLEN	PF Sideband Transmit Queue Length	13.2.2.22.157
0x0022FD80	PF_SB_ATQH	PF Sideband Transmit Head	13.2.2.22.158
0x0022FE00	PF_SB_ATQT	PF Sideband Transmit Tail	13.2.2.22.159
0x0022FE80	PF_SB_ARQBAL	PF Sideband Receive Queue Base Address Low	13.2.2.22.160
0x0022FF00	PF_SB_ARQBAH	PF Sideband Receive Queue Base Address High	13.2.2.22.161
0x0022FF80	PF_SB_ARQLEN	PF Sideband Receive Queue Length	13.2.2.22.162
0x00230000	PF_SB_ARQH	PF Sideband Receive Head	13.2.2.22.163

Table 13-30. PF - Control Queues Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00230080	PF_SB_ARQT	PF Sideband Receive Tail	13.2.2.22.164
0x002300C0	PF0_SB_HLP_ATQBAL	PF0 HLP Sideband Transmit Queue Base Address Low	13.2.2.22.165
0x002300C4	PF0_SB_HLP_ATQBAH	PF0 HLP Sideband Transmit Queue Base Address High	13.2.2.22.166
0x002300C8	PF0_SB_HLP_ATQLEN	PF0 HLP Sideband Transmit Queue Length	13.2.2.22.167
0x002300CC	PF0_SB_HLP_ATQH	PF0 HLP Sideband Transmit Head	13.2.2.22.168
0x002300D0	PF0_SB_HLP_ATQT	PF0 HLP Sideband Transmit Tail	13.2.2.22.169
0x002300D4	PF0_SB_HLP_ARQBAL	PF0 HLP Sideband Receive Queue Base Address Low	13.2.2.22.170
0x002300D8	PF0_SB_HLP_ARQBAH	PF0 HLP Sideband Receive Queue Base Address High	13.2.2.22.171
0x002300DC	PF0_SB_HLP_ARQLEN	PF0 HLP Sideband Receive Queue Length	13.2.2.22.172
0x002300E0	PF0_SB_HLP_ARQH	PF0 HLP Sideband Receive Head	13.2.2.22.173
0x002300E4	PF0_SB_HLP_ARQT	PF0 HLP Sideband Receive Tail	13.2.2.22.174
0x002300E8	PF0_SB_HLP_REM_DEV_CTL	PF SB HLP Remote Device Control Register	13.2.2.22.175
0x002300EC	VF_SB_CPM_REM_DEV_CTL	VF SB CPM Remote Device Control Register	13.2.2.22.176
0x002300F0	PF_SB_REM_DEV_CTL	PF SB Remote Device Control Register	13.2.2.22.177
0x002300F4	PF0_SB_CPM_REM_DEV_CTL	PF SB CPM Remote Device Control Register	13.2.2.22.178
0x002300F8 + 0x4*n, n=0...7	SB_REM_DEV_DEST[n]	SB Remote Device Destination Register	13.2.2.22.179
0x00230800 + 0x4*VSI, VSI=0...767	VP_MBX_PF_VF_CTRL[VSI]	PF VF Control Register	13.2.2.22.180
0x00231800 + 0x4*VP128, VP128=0...127	VP_MBX_CPM_PF_VF_CTRL[VP128]	PF VF Control Register	13.2.2.22.181
0x00231A00 + 0x4*VP16, VP16=0...15	VP_MBX_HLP_PF_VF_CTRL[VP16]	PF VF Control Register	13.2.2.22.182
0x00231A40 + 0x4*VP16, VP16=0...15	VP_MBX_PSM_PF_VF_CTRL[VP16]	PF VF Control Register	13.2.2.22.183
0x00231C00 + 0x4*VP128, VP128=0...127	VP_SB_CPM_PF_VF_CTRL[VP128]	PF VF Control Register	13.2.2.22.184
0x00231EC0	GL_MBX_PASID	PF VF Control Register	13.2.2.22.185

Table 13-31. PF - Statistics Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00099094 + 0x4*n, n=0...63	TPB_PRTTPB_STAT_TC_BYTES_SENT[n]	PORT TC Transmit Byte Count	13.2.2.23.1
0x00099470 + 0x4*n, n=0...7	TPB_PRTTPB_STAT_PKT_SENT[n]	PORT Transmit Packet Count	13.2.2.23.2
0x000AC260	PRTRPB_RDPC	Port (Line) Receive Drop Counter	13.2.2.23.3
0x000AC280	PRTRPB_LDPC	Port (LB) Receive Drop Counter	13.2.2.23.4

Table 13-31. PF - Statistics Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00294C04 + 0x4*n, n=0...767	GLV_RDPC[n]	VSI Received Discard Packet Count	13.2.2.23.5
0x00295804 + 0x4*n, n=0...767	GLV_REPC[n]	Per VSI Error Drops	13.2.2.23.6
0x00300000 + 0x8*n, n=0...767	GLV_GOTCL[n]	VSI Good Octets Transmit Count Low	13.2.2.23.7
0x00300004 + 0x8*n, n=0...767	GLV_GOTCH[n]	VSI Good Octets Transmit Count High	13.2.2.23.8
0x00302000 + 0x8*n, n=0...31	GLSW_GOTCL[n]	Switch Good Octets Transmit Count Low	13.2.2.23.9
0x00302004 + 0x8*n, n=0...31	GLSW_GOTCH[n]	Switch Good Octets Transmit Count High	13.2.2.23.10
0x00304000 + 0x8*n, n=0...127	GL_STAT_SWR_GOTCL[n]	VEB VLAN Transmit Byte Count Low	13.2.2.23.11
0x00304004 + 0x8*n, n=0...127	GL_STAT_SWR_GOTCH[n]	VEB VLAN Transmit Byte Count High	13.2.2.23.12
0x00306000 + 0x8*n + 0x40*m, n=0...7, m=0...31	GLVEBUP_TBCL[n,m]	VEB UP Transmit Byte Count Low	13.2.2.23.13
0x00306004 + 0x8*n + 0x40*m, n=0...7, m=0...31	GLVEBUP_TBCH[n,m]	VEB UP Transmit Byte Count High	13.2.2.23.14
0x00308000 + 0x8*n + 0x40*m, n=0...7, m=0...31	GLVEBUP_TPCL[n,m]	VEB UP Transmit Packet Count Low	13.2.2.23.15
0x00308004 + 0x8*n + 0x40*m, n=0...7, m=0...31	GLVEBUP_TPCH[n,m]	VEB UP Transmit Packet Count High	13.2.2.23.16
0x0030A000 + 0x8*n, n=0...767	GLV_UPTCL[n]	VSI Unicast Packets Transmit Count Low	13.2.2.23.17
0x0030A004 + 0x8*n, n=0...767	GLV_UPTCH[n]	VSI Unicast Packets Transmit Count High	13.2.2.23.18
0x0030C000 + 0x8*n, n=0...767	GLV_MPTCL[n]	VSI Multicast Packets Transmit Count Low	13.2.2.23.19
0x0030C004 + 0x8*n, n=0...767	GLV_MPTCH[n]	VSI Multicast Packets Transmit Count High	13.2.2.23.20
0x0030E000 + 0x8*n, n=0...767	GLV_BPTCL[n]	VSI Broadcast Packets Transmit Count Low	13.2.2.23.21
0x0030E004 + 0x8*n, n=0...767	GLV_BPTCH[n]	VSI Broadcast Packets Transmit Count High	13.2.2.23.22
0x00310000 + 0x8*n, n=0...31	GLSW_UPTCL[n]	Switch Unicast Packets Transmit Count Low	13.2.2.23.23
0x00310004 + 0x8*n, n=0...31	GLSW_UPTCH[n]	Switch Unicast Packets Transmit Count High	13.2.2.23.24
0x00310100 + 0x8*n, n=0...31	GLSW_MPTCL[n]	Switch Multicast Packets Transmit Count Low	13.2.2.23.25
0x00310104 + 0x8*n, n=0...31	GLSW_MPTCH[n]	Switch Multicast Packets Transmit Count High	13.2.2.23.26
0x00310200 + 0x8*n, n=0...31	GLSW_BPTCL[n]	Switch Broadcast Packets Transmit Count Low	13.2.2.23.27
0x00310204 + 0x8*n, n=0...31	GLSW_BPTCH[n]	Switch Broadcast Packets Transmit Count High	13.2.2.23.28

Table 13-31. PF - Statistics Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00312000 + 0x4*VSI, VSI=0...767	GLV_TEPC[VSI]	VSI Transmit Error Packet Count	13.2.2.23.29
0x00340000 + 0x4*n, n=0...7	GLPRT_STDC[n]	Port Storm Control Discarded Count	13.2.2.23.30
0x00341000 + 0x8*n, n=0...31	GLSW_GORCL[n]	Switch Good Octets Received Count Low	13.2.2.23.31
0x00341004 + 0x8*n, n=0...31	GLSW_GORCH[n]	Switch Good Octets Received Count High	13.2.2.23.32
0x00342000 + 0x8*n, n=0...127	GL_STAT_SWR_GORCL[n]	VEB VLAN Receive Byte Count Low	13.2.2.23.33
0x00342004 + 0x8*n, n=0...127	GL_STAT_SWR_GORCH[n]	VEB VLAN Receive Byte Count High	13.2.2.23.34
0x00343000 + 0x8*n + 0x40*m, n=0...7, m=0...31	GLVEBUP_RBCL[n,m]	VEB UP Receive Byte Count Low	13.2.2.23.35
0x00343004 + 0x8*n + 0x40*m, n=0...7, m=0...31	GLVEBUP_RBCH[n,m]	VEB UP Receive Byte Count High	13.2.2.23.36
0x00344000 + 0x8*n + 0x40*m, n=0...7, m=0...31	GLVEBUP_RPCL[n,m]	VEB UP Receive Packet Count Low	13.2.2.23.37
0x00344004 + 0x8*n + 0x40*m, n=0...7, m=0...31	GLVEBUP_RPCH[n,m]	VEB UP Receive Packet Count High	13.2.2.23.38
0x00346000 + 0x8*n, n=0...31	GLSW_UPRCL[n]	Switch Unicast Packets Received Count Low	13.2.2.23.39
0x00346004 + 0x8*n, n=0...31	GLSW_UPRCH[n]	Switch Unicast Packets Received Count High	13.2.2.23.40
0x00346100 + 0x8*n, n=0...31	GLSW_MPRCL[n]	Switch Multicast Packets Received Count Low	13.2.2.23.41
0x00346104 + 0x8*n, n=0...31	GLSW_MPRCH[n]	Switch Multicast Packets Received Count High	13.2.2.23.42
0x00346200 + 0x8*n, n=0...31	GLSW_BPRCL[n]	Switch Broadcast Packets Received Count Low	13.2.2.23.43
0x00346204 + 0x8*n, n=0...31	GLSW_BPRCH[n]	Switch Broadcast Packets Received Count High	13.2.2.23.44
0x00347000 + 0x8*n, n=0...127	GL_STAT_SWR_UPCL[n]	VEB VLAN Unicast Packet Count Low	13.2.2.23.45
0x00347004 + 0x8*n, n=0...127	GL_STAT_SWR_UPCH[n]	VEB VLAN Unicast Packet Count High	13.2.2.23.46
0x00347400 + 0x8*n, n=0...127	GL_STAT_SWR_MPCL[n]	VEB VLAN Multicast Packet Count Low	13.2.2.23.47
0x00347404 + 0x8*n, n=0...127	GL_STAT_SWR_MPCH[n]	VEB VLAN Multicast Packet Count High	13.2.2.23.48
0x00347800 + 0x8*n, n=0...127	GL_STAT_SWR_BPCL[n]	VEB VLAN Broadcast Packet Count Low	13.2.2.23.49
0x00347804 + 0x8*n, n=0...127	GL_STAT_SWR_BPCH[n]	VEB VLAN Broadcast Packet Count High	13.2.2.23.50
0x00380000 + 0x8*n, n=0...7	GLPRT_GORCL[n]	Port Good Octets Received Count Low	13.2.2.23.51
0x00380004 + 0x8*n, n=0...7	GLPRT_GORCH[n]	Port Good Octets Received Count High	13.2.2.23.52

Table 13-31. PF - Statistics Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00380040 + 0x8*n, n=0...7	GLPRT_MLFC[n]	Port MAC Local Fault Count	13.2.2.23.53
0x00380044 + 0x8*n, n=0...7	GLPRT_MLFC_H[n]	Port MAC Local Fault Count	13.2.2.23.54
0x00380080 + 0x8*n, n=0...7	GLPRT_MRFC[n]	Port MAC Remote Fault Count	13.2.2.23.55
0x00380084 + 0x8*n, n=0...7	GLPRT_MRFC_H[n]	Port MAC Remote Fault Count	13.2.2.23.56
0x00380100 + 0x8*n, n=0...7	GLPRT_CRCERRS[n]	Port CRC Error Count	13.2.2.23.57
0x00380104 + 0x8*n, n=0...7	GLPRT_CRCERRS_H[n]	Port CRC Error Count	13.2.2.23.58
0x00380140 + 0x8*n, n=0...7	GLPRT_RLEC[n]	Receive Length Error Count	13.2.2.23.59
0x00380144 + 0x8*n, n=0...7	GLPRT_RLEC_H[n]	Receive Length Error Count	13.2.2.23.60
0x003801C0 + 0x8*n, n=0...7	GLPRT_ILLERRC[n]	Port Illegal Byte Error Count	13.2.2.23.61
0x003801C4 + 0x8*n, n=0...7	GLPRT_ILLERRC_H[n]	Port Illegal Byte Error Count	13.2.2.23.62
0x00380200 + 0x8*n, n=0...7	GLPRT_RUC[n]	Receive Undersize Count	13.2.2.23.63
0x00380204 + 0x8*n, n=0...7	GLPRT_RUC_H[n]	Receive Undersize Count	13.2.2.23.64
0x00380240 + 0x8*n, n=0...7	GLPRT_ROC[n]	Receive Oversize Count	13.2.2.23.65
0x00380244 + 0x8*n, n=0...7	GLPRT_ROC_H[n]	Receive Oversize Count	13.2.2.23.66
0x00380280 + 0x8*n, n=0...7	GLPRT_LXONRXC[n]	Port Link XON Received Count	13.2.2.23.67
0x00380284 + 0x8*n, n=0...7	GLPRT_LXONRXC_H[n]	Port Link XON Received Count	13.2.2.23.68
0x003802C0 + 0x8*n, n=0...7	GLPRT_LXOFFRXC[n]	Port Link XOFF Received Count	13.2.2.23.69
0x003802C4 + 0x8*n, n=0...7	GLPRT_LXOFFRXC_H[n]	Port Link XOFF Received Count	13.2.2.23.70
0x00380300 + 0x8*n + 0x40*m, n=0...7, m=0...7	GLPRT_PXONRXC[n,m]	Priority XON Received Count	13.2.2.23.71
0x00380304 + 0x8*n + 0x40*m, n=0...7, m=0...7	GLPRT_PXONRXC_H[n,m]	Priority XON Received Count	13.2.2.23.72
0x00380500 + 0x8*n + 0x40*m, n=0...7, m=0...7	GLPRT_PXOFFRXC[n,m]	Priority XOFF Received Count	13.2.2.23.73
0x00380504 + 0x8*n + 0x40*m, n=0...7, m=0...7	GLPRT_PXOFFRXC_H[n,m]	Priority XOFF Received Count	13.2.2.23.74
0x00380700 + 0x8*n + 0x40*m, n=0...7, m=0...7	GLPRT_RXON2OFFCNT[n,m]	Priority XON to XOFF Count	13.2.2.23.75
0x00380704 + 0x8*n + 0x40*m, n=0...7, m=0...7	GLPRT_RXON2OFFCNT_H[n,m]	Priority XON to XOFF Count	13.2.2.23.76

Table 13-31. PF - Statistics Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00380900 + 0x8*n, n=0...7	GLPRT_PRC64L[n]	Packets Received [64 Bytes] Count Low	13.2.2.23.77
0x00380904 + 0x8*n, n=0...7	GLPRT_PRC64H[n]	Packets Received [64 Bytes] Count High	13.2.2.23.78
0x00380940 + 0x8*n, n=0...7	GLPRT_PRC127L[n]	Packets Received [65-127 Bytes] Count Low	13.2.2.23.79
0x00380944 + 0x8*n, n=0...7	GLPRT_PRC127H[n]	Packets Received [65-127 Bytes] Count High	13.2.2.23.80
0x00380980 + 0x8*n, n=0...7	GLPRT_PRC255L[n]	Packets Received [128-255 Bytes] Count Low	13.2.2.23.81
0x00380984 + 0x8*n, n=0...7	GLPRT_PRC255H[n]	Packets Received [128-255 Bytes] Count High	13.2.2.23.82
0x003809C0 + 0x8*n, n=0...7	GLPRT_PRC511L[n]	Packets Received [256-511 Bytes] Count Low	13.2.2.23.83
0x003809C4 + 0x8*n, n=0...7	GLPRT_PRC511H[n]	Packets Received [256-511 Bytes] Count High	13.2.2.23.84
0x00380A00 + 0x8*n, n=0...7	GLPRT_PRC1023L[n]	Packets Received [512-1023 Bytes] Count Low	13.2.2.23.85
0x00380A04 + 0x8*n, n=0...7	GLPRT_PRC1023H[n]	Packets Received [512-1023 Bytes] Count High	13.2.2.23.86
0x00380A40 + 0x8*n, n=0...7	GLPRT_PRC1522L[n]	Packets Received [1024-1522 Bytes] Count Low	13.2.2.23.87
0x00380A44 + 0x8*n, n=0...7	GLPRT_PRC1522H[n]	Packets Received [1024-1522 Bytes] Count High	13.2.2.23.88
0x00380A80 + 0x8*n, n=0...7	GLPRT_PRC9522L[n]	Packets Received [1523-9522 Bytes] Count Low	13.2.2.23.89
0x00380A84 + 0x8*n, n=0...7	GLPRT_PRC9522H[n]	Packets Received [1523-9522 Bytes] Count High	13.2.2.23.90
0x00380AC0 + 0x8*n, n=0...7	GLPRT_RFC[n]	Receive Fragment Count	13.2.2.23.91
0x00380AC4 + 0x8*n, n=0...7	GLPRT_RFC_H[n]	Receive Fragment Count	13.2.2.23.92
0x00380B00 + 0x8*n, n=0...7	GLPRT_RJC[n]	Receive Jabber Count	13.2.2.23.93
0x00380B04 + 0x8*n, n=0...7	GLPRT_RJC_H[n]	Receive Jabber Count	13.2.2.23.94
0x00380B40 + 0x8*n, n=0...7	GLPRT_GOTCL[n]	Port Good Octets Transmit Count Low	13.2.2.23.95
0x00380B44 + 0x8*n, n=0...7	GLPRT_GOTCH[n]	Port Good Octets Transmit Count High	13.2.2.23.96
0x00380B80 + 0x8*n, n=0...7	GLPRT_PTC64L[n]	Packets Transmitted [64 Bytes] Count Low	13.2.2.23.97
0x00380B84 + 0x8*n, n=0...7	GLPRT_PTC64H[n]	Packets Transmitted [64 Bytes] Count High	13.2.2.23.98
0x00380BC0 + 0x8*n, n=0...7	GLPRT_PTC127L[n]	Packets Transmitted [65-127 Bytes] Count Low	13.2.2.23.99
0x00380BC4 + 0x8*n, n=0...7	GLPRT_PTC127H[n]	Packets Transmitted [65-127 Bytes] Count High	13.2.2.23.100

Table 13-31. PF - Statistics Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00380C00 + 0x8*n, n=0...7	GLPRT_PTC255L[n]	Packets Transmitted [128-255 Bytes] Count Low	13.2.2.23.101
0x00380C04 + 0x8*n, n=0...7	GLPRT_PTC255H[n]	Packets Transmitted [128-255 Bytes] Count High	13.2.2.23.102
0x00380C40 + 0x8*n, n=0...7	GLPRT_PTC511L[n]	Packets Transmitted [256-511 Bytes] Count Low	13.2.2.23.103
0x00380C44 + 0x8*n, n=0...7	GLPRT_PTC511H[n]	Packets Transmitted [256-511 Bytes] Count High	13.2.2.23.104
0x00380C80 + 0x8*n, n=0...7	GLPRT_PTC1023L[n]	Packets Transmitted [512-1023 Bytes] Count Low	13.2.2.23.105
0x00380C84 + 0x8*n, n=0...7	GLPRT_PTC1023H[n]	Packets Transmitted [512-1023 Bytes] Count High	13.2.2.23.106
0x00380CC0 + 0x8*n, n=0...7	GLPRT_PTC1522L[n]	Packets Transmitted [1024-1522 Bytes] Count Low	13.2.2.23.107
0x00380CC4 + 0x8*n, n=0...7	GLPRT_PTC1522H[n]	Packets Transmitted [1024-1522 Bytes] Count High	13.2.2.23.108
0x00380D00 + 0x8*n, n=0...7	GLPRT_PTC9522L[n]	Packets Transmitted [1523-9522 Bytes] Count Low	13.2.2.23.109
0x00380D04 + 0x8*n, n=0...7	GLPRT_PTC9522H[n]	Packets Transmitted [1523-9522 Bytes] Count High	13.2.2.23.110
0x00380D40 + 0x8*n + 0x40*m, n=0...7, m=0...7	GLPRT_PXONTXC[n,m]	Priority XON Transmitted Count	13.2.2.23.111
0x00380D44 + 0x8*n + 0x40*m, n=0...7, m=0...7	GLPRT_PXONTXC_H[n,m]	Priority XON Transmitted Count	13.2.2.23.112
0x00380F40 + 0x8*n + 0x40*m, n=0...7, m=0...7	GLPRT_PXOFFTXC[n,m]	Priority XOFF Transmitted Count	13.2.2.23.113
0x00380F44 + 0x8*n + 0x40*m, n=0...7, m=0...7	GLPRT_PXOFFTXC_H[n,m]	Priority XOFF Transmitted Count	13.2.2.23.114
0x00381140 + 0x8*n, n=0...7	GLPRT_LXONTXC[n]	Port Link XON Transmitted Count	13.2.2.23.115
0x00381144 + 0x8*n, n=0...7	GLPRT_LXONTXC_H[n]	Port Link XON Transmitted Count	13.2.2.23.116
0x00381180 + 0x8*n, n=0...7	GLPRT_LXOFFTXC[n]	Port Link XOFF Transmitted Count	13.2.2.23.117
0x00381184 + 0x8*n, n=0...7	GLPRT_LXOFFTXC_H[n]	Port Link XOFF Transmitted Count	13.2.2.23.118
0x003811C0 + 0x8*n, n=0...7	GLPRT_UPTCL[n]	Port Unicast Packets Transmit Count Low	13.2.2.23.119
0x003811C4 + 0x8*n, n=0...7	GLPRT_UPTCH[n]	Port Unicast Packets Transmit Count High	13.2.2.23.120
0x00381200 + 0x8*n, n=0...7	GLPRT_MPTCL[n]	Port Multicast Packets Transmit Count Low	13.2.2.23.121
0x00381204 + 0x8*n, n=0...7	GLPRT_MPTCH[n]	Port Multicast Packets Transmit Count High	13.2.2.23.122
0x00381240 + 0x8*n, n=0...7	GLPRT_BPTCL[n]	Port Broadcast Packets Transmit Count Low	13.2.2.23.123
0x00381244 + 0x8*n, n=0...7	GLPRT_BPTCH[n]	Port Broadcast Packets Transmit Count High	13.2.2.23.124

Table 13-31. PF - Statistics Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00381280 + 0x8*n, n=0...7	GLPRT_TDOLD[n]	Transmit Discard on Link Down	13.2.2.23.125
0x00381284 + 0x8*n, n=0...7	GLPRT_TDOLD_H[n]	Transmit Discard On Link Down	13.2.2.23.126
0x00381300 + 0x8*n, n=0...7	GLPRT_UPRCL[n]	Port Unicast Packets Received Count Low	13.2.2.23.127
0x00381304 + 0x8*n, n=0...7	GLPRT_UPRCH[n]	Port Unicast Packets Received Count High	13.2.2.23.128
0x00381340 + 0x8*n, n=0...7	GLPRT_MPRCL[n]	Port Multicast Packets Received Count Low	13.2.2.23.129
0x00381344 + 0x8*n, n=0...7	GLPRT_MPRCH[n]	Port Multicast Packets Received Count High	13.2.2.23.130
0x00381380 + 0x8*n, n=0...7	GLPRT_BPRCL[n]	Port Broadcast Packets Received Count Low	13.2.2.23.131
0x00381384 + 0x8*n, n=0...7	GLPRT_BPRCH[n]	Port Broadcast Packets Received Count High	13.2.2.23.132
0x00388000 + 0x8*n, n=0...511	GLSTAT_ACL_CNT_0_L[n]	ACL Counter Bank 0 LSBs	13.2.2.23.133
0x00388004 + 0x8*n, n=0...511	GLSTAT_ACL_CNT_0_H[n]	ACL Counter Bank 0 MSBs	13.2.2.23.134
0x00389000 + 0x8*n, n=0...511	GLSTAT_ACL_CNT_1_L[n]	ACL Counter Bank 1 LSBs	13.2.2.23.135
0x00389004 + 0x8*n, n=0...511	GLSTAT_ACL_CNT_1_H[n]	ACL Counter Bank 1 MSBs	13.2.2.23.136
0x0038A000 + 0x8*n, n=0...511	GLSTAT_ACL_CNT_2_L[n]	ACL Counter Bank 2 LSBs	13.2.2.23.137
0x0038A004 + 0x8*n, n=0...511	GLSTAT_ACL_CNT_2_H[n]	ACL Counter Bank 2 MSBs	13.2.2.23.138
0x0038B000 + 0x8*n, n=0...511	GLSTAT_ACL_CNT_3_L[n]	ACL Counter Bank 3 LSBs	13.2.2.23.139
0x0038B004 + 0x8*n, n=0...511	GLSTAT_ACL_CNT_3_H[n]	ACL Counter Bank 3 MSBs	13.2.2.23.140
0x003A0000 + 0x8*n, n=0...4095	GLSTAT_FD_CNT0L[n]	Global Packet Byte Statistic Counter Bank 0 Low	13.2.2.23.141
0x003A0004 + 0x8*n, n=0...4095	GLSTAT_FD_CNT0H[n]	Global Packet Byte Statistic Counter Bank 0 High	13.2.2.23.142
0x003A8000 + 0x8*n, n=0...4095	GLSTAT_FD_CNT1L[n]	Global Packet Byte Statistic Counter Bank 1 Low	13.2.2.23.143
0x003A8004 + 0x8*n, n=0...4095	GLSTAT_FD_CNT1H[n]	Global Packet Byte Statistic Counter Bank 1 High	13.2.2.23.144
0x003B0000 + 0x8*n, n=0...767	GLV_GORCL[n]	VSI Good Octets Received Count Low	13.2.2.23.145
0x003B0004 + 0x8*n, n=0...767	GLV_GORCH[n]	VSI Good Octets Received Count High	13.2.2.23.146
0x003B2000 + 0x8*n, n=0...767	GLV_UPRCL[n]	VSI Unicast Packets Received Count Low	13.2.2.23.147
0x003B2004 + 0x8*n, n=0...767	GLV_UPRCH[n]	VSI Unicast Packets Received Count High	13.2.2.23.148

Table 13-31. PF - Statistics Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x003B4000 + 0x8*n, n=0...767	GLV_MPRCL[n]	VSI Multicast Packets Received Count Low	13.2.2.23.149
0x003B4004 + 0x8*n, n=0...767	GLV_MPRCH[n]	VSI Multicast Packets Received Count High	13.2.2.23.150
0x003B6000 + 0x8*n, n=0...767	GLV_BPRCL[n]	VSI Broadcast Packets Received Count Low	13.2.2.23.151
0x003B6004 + 0x8*n, n=0...767	GLV_BPRCH[n]	VSI Broadcast Packets Received Count High	13.2.2.23.152

Table 13-32. PF - Protocol Engine Statistics Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00540000 + 0x4*n, n=0...127	GLPES_PFRXVLANERR[n]	Protocol Engine Statistics Received VLAN_ID Errors	13.2.2.24.1
0x00540400 + 0x8*n, n=0...127	GLPES_PFIP4RXOCTSLO[n]	Protocol Engine Statistics IPv4 Received Octets Low	13.2.2.24.2
0x00540404 + 0x8*n, n=0...127	GLPES_PFIP4RXOCTSHI[n]	Protocol Engine Statistics IPv4 Received Octets High	13.2.2.24.3
0x00540C00 + 0x8*n, n=0...127	GLPES_PFIP4RXPKTSLO[n]	Protocol Engine Statistics IPv4 Received Packets Low	13.2.2.24.4
0x00540C04 + 0x8*n, n=0...127	GLPES_PFIP4RXPKTSHI[n]	Protocol Engine Statistics IPv4 Received Packets High	13.2.2.24.5
0x00541400 + 0x4*n, n=0...127	GLPES_PFIP4RXDISCARD[n]	Protocol Engine Statistics IPv4 Discards	13.2.2.24.6
0x00541800 + 0x4*n, n=0...127	GLPES_PFIP4RXTRUNC[n]	Protocol Engine Statistics IPv4 Truncated Packets	13.2.2.24.7
0x00541C00 + 0x8*n, n=0...127	GLPES_PFIP4RXFRAGSLO[n]	Protocol Engine Statistics IPv4 Received Fragments Low	13.2.2.24.8
0x00541C04 + 0x8*n, n=0...127	GLPES_PFIP4RXFRAGSHI[n]	Protocol Engine Statistics IPv4 Received Fragments High	13.2.2.24.9
0x00542400 + 0x8*n, n=0...127	GLPES_PFIP4RXMCOCTSLO[n]	Protocol Engine Statistics IPv4 Received Multicast Octets Low	13.2.2.24.10
0x00542404 + 0x8*n, n=0...127	GLPES_PFIP4RXMCOCTSHI[n]	Protocol Engine Statistics IPv4 Received Multicast Octets High	13.2.2.24.11
0x00542C00 + 0x8*n, n=0...127	GLPES_PFIP4RXMCPKTSLO[n]	Protocol Engine Statistics IPv4 Received Multicast Packets Low	13.2.2.24.12
0x00542C04 + 0x8*n, n=0...127	GLPES_PFIP4RXMCPKTSHI[n]	Protocol Engine Statistics IPv4 Received Multicast Packets High	13.2.2.24.13
0x00543400 + 0x8*n, n=0...127	GLPES_PFIP6RXOCTSLO[n]	Protocol Engine Statistics IPv6 Received Octets Low	13.2.2.24.14
0x00543404 + 0x8*n, n=0...127	GLPES_PFIP6RXOCTSHI[n]	Protocol Engine Statistics IPv6 Received Octets High	13.2.2.24.15
0x00543C00 + 0x8*n, n=0...127	GLPES_PFIP6RXPKTSLO[n]	Protocol Engine Statistics IPv6 Received Packets Low	13.2.2.24.16
0x00543C04 + 0x8*n, n=0...127	GLPES_PFIP6RXPKTSHI[n]	Protocol Engine Statistics IPv6 Received Packets High	13.2.2.24.17
0x00544400 + 0x4*n, n=0...127	GLPES_PFIP6RXDISCARD[n]	Protocol Engine Statistics IPv6 Discards	13.2.2.24.18

Table 13-32. PF - Protocol Engine Statistics Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00544800 + 0x4*n, n=0...127	GLPES_PFIP6RXTRUNC[n]	Protocol Engine Statistics IPv6 Truncated Packets	13.2.2.24.19
0x00544C00 + 0x8*n, n=0...127	GLPES_PFIP6RXFRAGSLO[n]	Protocol Engine Statistics IPv6 Received Fragments Low	13.2.2.24.20
0x00544C04 + 0x8*n, n=0...127	GLPES_PFIP6RXFRAGSHI[n]	Protocol Engine Statistics IPv6 Received Fragments High	13.2.2.24.21
0x00545400 + 0x8*n, n=0...127	GLPES_PFIP6RXMCOCTSLO[n]	Protocol Engine Statistics IPv6 Received Multicast Octets Low	13.2.2.24.22
0x00545404 + 0x8*n, n=0...127	GLPES_PFIP6RXMCOCTSHI[n]	Protocol Engine Statistics IPv6 Received Multicast Octets High	13.2.2.24.23
0x00545C00 + 0x8*n, n=0...127	GLPES_PFIP6RXMCPKTSLO[n]	Protocol Engine Statistics IPv6 Received Multicast Packets Low	13.2.2.24.24
0x00545C04 + 0x8*n, n=0...127	GLPES_PFIP6RXMCPKTSHI[n]	Protocol Engine Statistics IPv6 Received Multicast Packets High	13.2.2.24.25
0x00546400 + 0x8*n, n=0...127	GLPES_PFIP4TXOCTSLO[n]	Protocol Engine Statistics IPv4 Transmitted Octets Low	13.2.2.24.26
0x00546404 + 0x8*n, n=0...127	GLPES_PFIP4TXOCTSHI[n]	Protocol Engine Statistics IPv4 Transmitted Octets High	13.2.2.24.27
0x00546C00 + 0x8*n, n=0...127	GLPES_PFIP4TXPKTSLO[n]	Protocol Engine Statistics IPv4 Transmitted Packets Low	13.2.2.24.28
0x00546C04 + 0x8*n, n=0...127	GLPES_PFIP4TXPKTSHI[n]	Protocol Engine Statistics IPv4 Transmitted Packets High	13.2.2.24.29
0x00547400 + 0x8*n, n=0...127	GLPES_PFIP4TXFRAGSLO[n]	Protocol Engine Statistics IPv4 Transmitted Fragments Low	13.2.2.24.30
0x00547404 + 0x8*n, n=0...127	GLPES_PFIP4TXFRAGSHI[n]	Protocol Engine Statistics IPv4 Transmitted Fragments High	13.2.2.24.31
0x00547C00 + 0x8*n, n=0...127	GLPES_PFIP4TXMCOCTSLO[n]	Protocol Engine Statistics IPv4 Transmitted Multicast Octets Low	13.2.2.24.32
0x00547C04 + 0x8*n, n=0...127	GLPES_PFIP4TXMCOCTSHI[n]	Protocol Engine Statistics IPv4 Transmitted Multicast Octets High	13.2.2.24.33
0x00548400 + 0x8*n, n=0...127	GLPES_PFIP4TXMCPKTSLO[n]	Protocol Engine Statistics IPv4 Transmitted Multicast Packets Low	13.2.2.24.34
0x00548404 + 0x8*n, n=0...127	GLPES_PFIP4TXMCPKTSHI[n]	Protocol Engine Statistics IPv4 Transmitted Multicast Packets High	13.2.2.24.35
0x00548C00 + 0x8*n, n=0...127	GLPES_PFIP6TXOCTSLO[n]	Protocol Engine Statistics IPv6 Transmitted Octets Low	13.2.2.24.36
0x00548C04 + 0x8*n, n=0...127	GLPES_PFIP6TXOCTSHI[n]	Protocol Engine Statistics IPv6 Transmitted Octets High	13.2.2.24.37
0x00549400 + 0x8*n, n=0...127	GLPES_PFIP6TXPKTSLO[n]	Protocol Engine Statistics IPv6 Transmitted Packets Low	13.2.2.24.38
0x00549404 + 0x8*n, n=0...127	GLPES_PFIP6TXPKTSHI[n]	Protocol Engine Statistics IPv6 Transmitted Packets High	13.2.2.24.39
0x00549C00 + 0x8*n, n=0...127	GLPES_PFIP6TXFRAGSLO[n]	Protocol Engine Statistics IPv6 Transmitted Fragments Low	13.2.2.24.40
0x00549C04 + 0x8*n, n=0...127	GLPES_PFIP6TXFRAGSHI[n]	Protocol Engine Statistics IPv6 Transmitted Fragments High	13.2.2.24.41
0x0054A400 + 0x8*n, n=0...127	GLPES_PFIP6TXMCOCTSLO[n]	Protocol Engine Statistics IPv6 Transmitted Multicast Octets Low	13.2.2.24.42

Table 13-32. PF - Protocol Engine Statistics Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0054A404 + 0x8*n, n=0...127	GLPES_PFIP6TXMCOCTSHI[n]	Protocol Engine Statistics IPv6 Transmitted Multicast Octets High	13.2.2.24.43
0x0054AC00 + 0x8*n, n=0...127	GLPES_PFIP6TXMCPKTSLO[n]	Protocol Engine Statistics IPv6 Transmitted Multicast Packets Low	13.2.2.24.44
0x0054AC04 + 0x8*n, n=0...127	GLPES_PFIP6TXMCPKTSHI[n]	Protocol Engine Statistics IPv6 Transmitted Multicast Packets High	13.2.2.24.45
0x0054B400 + 0x4*n, n=0...127	GLPES_PFIP4TXNOROUTE[n]	Protocol Engine Statistics IPv4 Discarded No Route Packets	13.2.2.24.46
0x0054B800 + 0x4*n, n=0...127	GLPES_PFIP6TXNOROUTE[n]	Protocol Engine Statistics IPv6 Discarded No Route Packets	13.2.2.24.47
0x0054BC00 + 0x8*n, n=0...127	GLPES_PFTCPRXSEGSLO[n]	Protocol Engine Statistics TCP Received Segments Low	13.2.2.24.48
0x0054BC04 + 0x8*n, n=0...127	GLPES_PFTCPRXSEGSHI[n]	Protocol Engine Statistics TCP Received Segments High	13.2.2.24.49
0x0054C400 + 0x4*n, n=0...127	GLPES_PFTCPRXOPTERR[n]	Protocol Engine Statistics TCP Received Segments with Unsupported Options	13.2.2.24.50
0x0054C800 + 0x4*n, n=0...127	GLPES_PFTCPRXPROTOERR[n]	Protocol Engine Statistics TCP Dropped Segments due to Protocol Errors	13.2.2.24.51
0x0054CC00 + 0x8*n, n=0...127	GLPES_PFTCPTXSEGLO[n]	Protocol Engine Statistics TCP Transmitted Segments Low	13.2.2.24.52
0x0054CC04 + 0x8*n, n=0...127	GLPES_PFTCPTXSEGHI[n]	Protocol Engine Statistics TCP Transmitted Segments High	13.2.2.24.53
0x0054D400 + 0x8*n, n=0...127	GLPES_PFUDPRXPKTSLO[n]	Protocol Engine Statistics UDP Received Packets Low	13.2.2.24.54
0x0054D404 + 0x8*n, n=0...127	GLPES_PFUDPRXPKTSHI[n]	Protocol Engine Statistics UDP Received Packets High	13.2.2.24.55
0x0054DC00 + 0x8*n, n=0...127	GLPES_PFUDPTXPKTSLO[n]	Protocol Engine Statistics UDP Transmitted Packets Low	13.2.2.24.56
0x0054DC04 + 0x8*n, n=0...127	GLPES_PFUDPTXPKTSHI[n]	Protocol Engine Statistics UDP Transmitted Packets High	13.2.2.24.57
0x0054E400 + 0x8*n, n=0...127	GLPES_PFRDMARXWRSLO[n]	Protocol Engine Statistics RDMA Received Write Messages Low	13.2.2.24.58
0x0054E404 + 0x8*n, n=0...127	GLPES_PFRDMARXWRSHI[n]	Protocol Engine Statistics RDMA Received Write Messages High	13.2.2.24.59
0x0054EC00 + 0x8*n, n=0...127	GLPES_PFRDMARXRDSLO[n]	Protocol Engine Statistics RDMA Received Read Request Messages Low	13.2.2.24.60
0x0054EC04 + 0x8*n, n=0...127	GLPES_PFRDMARXRDSHI[n]	Protocol Engine Statistics RDMA Received Read Request Messages High	13.2.2.24.61
0x0054F400 + 0x8*n, n=0...127	GLPES_PFRDMARXSNDSLO[n]	Protocol Engine Statistics RDMA Received Send Messages Low	13.2.2.24.62
0x0054F404 + 0x8*n, n=0...127	GLPES_PFRDMARXSNDSHI[n]	Protocol Engine Statistics RDMA Received Send Messages High	13.2.2.24.63
0x0054FC00 + 0x8*n, n=0...127	GLPES_PFRDMATXWRSLO[n]	Protocol Engine Statistics RDMA Transmitted Write Messages Low	13.2.2.24.64
0x0054FC04 + 0x8*n, n=0...127	GLPES_PFRDMATXWRSHI[n]	Protocol Engine Statistics RDMA Transmitted Write Messages High	13.2.2.24.65
0x00550400 + 0x8*n, n=0...127	GLPES_PFRDMATXRDSLO[n]	Protocol Engine Statistics RDMA Transmitted Read Request Messages Low	13.2.2.24.66

Table 13-32. PF - Protocol Engine Statistics Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00550404 + 0x8*n, n=0...127	GLPES_PFRDMATXRDSHI[n]	Protocol Engine Statistics RDMA Transmitted Read Request Messages High	13.2.2.24.67
0x00550C00 + 0x8*n, n=0...127	GLPES_PFRDMATXSNDSLO[n]	Protocol Engine Statistics RDMA Transmitted Send Messages Low	13.2.2.24.68
0x00550C04 + 0x8*n, n=0...127	GLPES_PFRDMATXSNDSHI[n]	Protocol Engine Statistics RDMA Transmitted Send Messages High	13.2.2.24.69
0x00551400 + 0x8*n, n=0...127	GLPES_PFRDMAVBNDLO[n]	Protocol Engine Statistics RDMA Verbs Bind Operations Low	13.2.2.24.70
0x00551404 + 0x8*n, n=0...127	GLPES_PFRDMAVBNDHI[n]	Protocol Engine Statistics RDMA Verbs Bind Operations High	13.2.2.24.71
0x00551C00 + 0x8*n, n=0...127	GLPES_PFRDMAVINVLO[n]	Protocol Engine Statistics RDMA Verbs Invalidate Operations Low	13.2.2.24.72
0x00551C04 + 0x8*n, n=0...127	GLPES_PFRDMAVINVHI[n]	Protocol Engine Statistics RDMA Verbs Invalidate Operations High	13.2.2.24.73
0x00552400 + 0x4*n, n=0...127	GLPES_PFTCPRTXSEG[n]	Protocol Engine Statistics TCP Retransmitted Segments	13.2.2.24.74
0x00552800 + 0x4*n, n=0...127	GLPES_PFRXRPCNPIGNORED[n]	Protocol Engine Statistics Congestion Notification Packets Ignored	13.2.2.24.75
0x00552C00 + 0x4*n, n=0...127	GLPES_PFRXRPCNPHANDLED[n]	Protocol Engine Statistics Congestion Notification Packets Handled	13.2.2.24.76
0x00553000 + 0x8*n, n=0...127	GLPES_PFRXNPECNMARKEDPKTSLO[n]	Protocol Engine Statistics with ECN Bits Indicating Congestion Low	13.2.2.24.77
0x00553004 + 0x8*n, n=0...127	GLPES_PFRXNPECNMARKEDPKTSHI[n]	Protocol Engine Statistics with ECN Bits Indicating Congestion High	13.2.2.24.78
0x00553800 + 0x4*n, n=0...127	GLPES_PFTXNPCNPSENT[n]	Protocol Engine Congestion Indication Sent Count	13.2.2.24.79
0x0055E000	GLPES_RDMARXUNALIGN	Protocol Engine Statistics RDMA Received Unaligned FPDUs	13.2.2.24.80
0x0055E004	GLPES_RDMARXOOONOMARK	Protocol Engine Statistics RDMA Received Out of Order No Markers FPDUs	13.2.2.24.81
0x0055E008	GLPES_RDMARXMULTFPDUSLO	Protocol Engine Statistics RDMA Received Multiple FPDUs Low	13.2.2.24.82
0x0055E00C	GLPES_RDMARXMULTFPDUSHI	Protocol Engine Statistics RDMA Received Multiple FPDUs High	13.2.2.24.83
0x0055E010	GLPES_RDMARXOOODDPLO	Protocol Engine Statistics RDMA Out of Order Placed DDP Segments Low	13.2.2.24.84
0x0055E014	GLPES_RDMARXOOODDPHI	Protocol Engine Statistics RDMA Out of Order Placed DDP Segments High	13.2.2.24.85
0x0055E018	GLPES_TCPRXPUREACKSLO	Protocol Engine Statistics TCP Received Pure Acks Low	13.2.2.24.86
0x0055E01C	GLPES_TCPRXPUREACKHI	Protocol Engine Statistics TCP Received Pure Acks High	13.2.2.24.87
0x0055E020	GLPES_TCPRXONEHOLELO	Protocol Engine Statistics TCP Receive First Hole Low	13.2.2.24.88
0x0055E024	GLPES_TCPRXONEHOLEHI	Protocol Engine Statistics TCP Received First Hole High	13.2.2.24.89
0x0055E028	GLPES_TCPRXTWOHOLELO	Protocol Engine Statistics TCP Receive Second Hole Low	13.2.2.24.90

Table 13-32. PF - Protocol Engine Statistics Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0055E02C	GLPES_TCPRXTWOHOLEHI	Protocol Engine Statistics TCP Received Second Hole High	13.2.2.24.91
0x0055E030	GLPES_TCPRXTHREEHOLELO	Protocol Engine Statistics TCP Receive Third Hole Low	13.2.2.24.92
0x0055E034	GLPES_TCPRXTHREEHOLEHI	Protocol Engine Statistics TCP Received Third Hole High	13.2.2.24.93
0x0055E038	GLPES_TCPRXFOURHOLELO	Protocol Engine Statistics TCP Receive Fourth Hole Low	13.2.2.24.94
0x0055E03C	GLPES_TCPRXFOURHOLEHI	Protocol Engine Statistics TCP Receive Fourth Hole High	13.2.2.24.95
0x0055E040	GLPES_TCPTXRETRANSFASTLO	Protocol Engine Statistics TCP Fast Retransmissions Low	13.2.2.24.96
0x0055E044	GLPES_TCPTXRETRANSFASTHI	Protocol Engine Statistics TCP Fast Retransmissions High	13.2.2.24.97
0x0055E048	GLPES_TCPTXTOUTSFASTLO	Protocol Engine Statistics TCP Fast Retransmission Timeouts Low	13.2.2.24.98
0x0055E04C	GLPES_TCPTXTOUTSFASTHI	Protocol Engine Statistics TCP Fast Retransmissions Timeouts High	13.2.2.24.99
0x0055E050	GLPES_TCPTXTOUTSLO	Protocol Engine Statistics TCP Retransmission Timeouts Low	13.2.2.24.100
0x0055E054	GLPES_TCPTXTOUTSHI	Protocol Engine Statistics TCP Retransmissions Timeouts High	13.2.2.24.101

Table 13-33. PF - Comm Transmit Queues Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x000E0000 + 0x4*DBQM, DBQM=0...16383	QTX_COMM_HEAD[DBQM]	Global Transmit Queue Head	13.2.2.25.1
0x000F0000 + 0x4*CQ, CQ=0...511	GLCOMM_CQ_CTL[CQ]	Transmit Comm Scheduler Completion Queue Control	13.2.2.25.2
0x000F0800 + 0x4*CQ, CQ=0...511	GLTCLAN_CQ_CNTX0[CQ]	Tx Completion Queue Context Register 0	13.2.2.25.3
0x000F1000 + 0x4*CQ, CQ=0...511	GLTCLAN_CQ_CNTX1[CQ]	Tx Completion Queue Context Register 1	13.2.2.25.4
0x000F1800 + 0x4*CQ, CQ=0...511	GLTCLAN_CQ_CNTX2[CQ]	Tx Completion Queue Context Register 2	13.2.2.25.5
0x000F2000 + 0x4*CQ, CQ=0...511	GLTCLAN_CQ_CNTX3[CQ]	Tx Completion Queue Context Register 3	13.2.2.25.6
0x000F2800 + 0x4*CQ, CQ=0...511	GLTCLAN_CQ_CNTX4[CQ]	Tx Completion Queue Context Register 4	13.2.2.25.7
0x000F3000 + 0x4*CQ, CQ=0...511	GLTCLAN_CQ_CNTX5[CQ]	Tx Completion Queue Context Register 5	13.2.2.25.8
0x000F3800 + 0x4*CQ, CQ=0...511	GLTCLAN_CQ_CNTX6[CQ]	Tx Completion Queue Context Register 6	13.2.2.25.9
0x000F4000 + 0x4*CQ, CQ=0...511	GLTCLAN_CQ_CNTX7[CQ]	Tx Completion Queue Context Register 7	13.2.2.25.10
0x000F4800 + 0x4*CQ, CQ=0...511	GLTCLAN_CQ_CNTX8[CQ]	Tx Completion Queue Context Register 8	13.2.2.25.11

Table 13-33. PF - Comm Transmit Queues Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x000F5000 + 0x4*CQ, CQ=0...511	GLTCLAN_CQ_CNTX9[CQ]	Tx Completion Queue Context Register 9	13.2.2.25.12
0x000F5800 + 0x4*CQ, CQ=0...511	GLTCLAN_CQ_CNTX10[CQ]	Tx Completion Queue Context Register 10	13.2.2.25.13
0x000F6000 + 0x4*CQ, CQ=0...511	GLTCLAN_CQ_CNTX11[CQ]	Tx Completion Queue Context Register 11	13.2.2.25.14
0x000F6800 + 0x4*CQ, CQ=0...511	GLTCLAN_CQ_CNTX12[CQ]	Tx Completion Queue Context Register 12	13.2.2.25.15
0x000F7000 + 0x4*CQ, CQ=0...511	GLTCLAN_CQ_CNTX13[CQ]	Tx Completion Queue Context Register 13	13.2.2.25.16
0x000F7800 + 0x4*CQ, CQ=0...511	GLTCLAN_CQ_CNTX14[CQ]	Tx Completion Queue Context Register 14	13.2.2.25.17
0x000F8000 + 0x4*CQ, CQ=0...511	GLTCLAN_CQ_CNTX15[CQ]	Tx Completion Queue Context Register 15	13.2.2.25.18
0x000F8800 + 0x4*CQ, CQ=0...511	GLTCLAN_CQ_CNTX16[CQ]	Tx Completion Queue Context Register 16	13.2.2.25.19
0x000F9000 + 0x4*CQ, CQ=0...511	GLTCLAN_CQ_CNTX17[CQ]	Tx Completion Queue Context Register 17	13.2.2.25.20
0x000F9800 + 0x4*CQ, CQ=0...511	GLTCLAN_CQ_CNTX18[CQ]	Tx Completion Queue Context Register 18	13.2.2.25.21
0x000FA000 + 0x4*CQ, CQ=0...511	GLTCLAN_CQ_CNTX19[CQ]	Tx Completion Queue Context Register 19	13.2.2.25.22
0x000FA800 + 0x4*CQ, CQ=0...511	GLTCLAN_CQ_CNTX20[CQ]	Tx Completion Queue Context Register 20	13.2.2.25.23
0x000FB000 + 0x4*CQ, CQ=0...511	GLTCLAN_CQ_CNTX21[CQ]	Tx Completion Queue Context Register 21	13.2.2.25.24
0x000FC064	GLCOMM_MIN_MAX_PKT	Global Transmit Comm Min/Max Packet	13.2.2.25.25
0x000FC0B8	GLLAN_TCLAN_CACHE_CTL	Transmit Comm Scheduler Tx LAN Cache Control	13.2.2.25.26
0x002C0000 + 0x4*DBQM, DBQM=0...16383	QTX_COMM_DBELL[DBQM]	Transmit Comm Scheduler Queue Doorbell	13.2.2.25.27
0x002D0000 + 0x400*n + 0x4*DBLQ, n=0...4, DBLQ=0...255	QTX_COMM_DBLQ_CNTX[n,DBLQ]	Transmit Comm Scheduler Queue Context	13.2.2.25.28
0x002D1400 + 0x4*DBLQ, DBLQ=0...255	QTX_COMM_DBLQ_DBELL[DBLQ]	Transmit Comm Scheduler Queue Doorbell	13.2.2.25.29
0x002D2D40 + 0x4*n, n=0...9	GLCOMM_QTX_CNTX_DATA[n]	Transmit Comm Scheduler Queue Context Data	13.2.2.25.30
0x002D2D68 + 0x4*n, n=0...15	GLCOMM_QUANTA_PROF[n]	Global Transmit Comm Scheduler Quanta Profile	13.2.2.25.31
0x002D2DA8 + 0x4*n, n=0...7	GLCOMM_PKT_SHAPER_PROF[n]	Global Transmit Comm Scheduler Quanta Profile	13.2.2.25.32
0x002D2DC8	GLCOMM_QTX_CNTX_CTL	Transmit Comm Scheduler Queue Context Control	13.2.2.25.33
0x002D2DCC	GLCOMM_QTX_CNTX_STAT	Transmit Comm Scheduler Queue Context Status	13.2.2.25.34

Table 13-34. PF - LAN Transmit/Receive Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00040BA0	PRT_TDPUL2TAGSEN	L2 Tag - Enable	13.2.2.26.1
0x00049240 + 0x4*n, n=0...20	GL_HLP_PRT_IPG_PREAMBLE_SIZE[n]	Transmit DCSP to TC Enforcement - IPv4	13.2.2.26.2
0x00049294 + 0x4*n, n=0...3	GL_TDPU_PSM_DEFAULT_RECIPE[n]	Transmit TDPU Scheduler 4 Adjustment Default Recipe	13.2.2.26.3
0x00049308	GLLAN_TSOMSK_F	Global TSO TCP Mask First	13.2.2.26.4
0x0004930C	GLLAN_TSOMSK_M	Global TSO TCP Mask Middle	13.2.2.26.5
0x00049310	GLLAN_TSOMSK_L	Global TSO TCP Mask Last	13.2.2.26.6
0x00060000 + 0x800*n + 0x4*VF, n=0...15, VF=0...255	VPLAN_RX_QTABLE[n,VF]	VF PF Rx-Queue Mapping Table	13.2.2.26.7
0x00070000 + 0x800*n + 0x4*VF, n=0...3, VF=0...255	VPLAN_DB_QTABLE[n,VF]	VF PF DB Queue Mapping Table	13.2.2.26.8
0x00072000 + 0x4*VF, VF=0...255	VPLAN_RX_QBASE[VF]	VF PF Rx-Queue Range	13.2.2.26.9
0x00073000 + 0x4*VF, VF=0...255	VPLAN_RXQ_MAPENA[VF]	VF LAN RXQ Enablement	13.2.2.26.10
0x00073800 + 0x4*VF, VF=0...255	VPLAN_TXQ_MAPENA[VF]	VF LAN TXQ Enablement	13.2.2.26.11
0x00074C00 + 0x40*n + 0x4*VP16, n=0...15, VP16=0...15	VPDSI_RX_QTABLE[n,VP16]	VF PF Rx-Queue Mapping Table	13.2.2.26.12
0x00075680	PFLAN_DB_QALLOC	PF Doorbell Queue Allocation	13.2.2.26.13
0x00075700	PFLAN_CP_QALLOC	PF Completion Queue Allocation	13.2.2.26.14
0x00120000 + 0x4*QRX, QRX=0...2047	QRX_CTRL[QRX]	Global Receive Queue Control	13.2.2.26.15
0x001C0000 + 0x800*n + 0x4*VF, n=0...15, VF=0...255	VPLAN_TX_QTABLE[n,VF]	VF PF Tx-Queue Mapping Table	13.2.2.26.16
0x001D1800 + 0x4*VF, VF=0...255	VPLAN_TX_QBASE[VF]	VF PF Tx-Queue Range	13.2.2.26.17
0x001D2000 + 0x40*n + 0x4*VP16, n=0...15, VP16=0...15	VPDSI_TX_QTABLE[n,VP16]	VF PF Tx-Queue Mapping Table	13.2.2.26.18
0x001D2500	PFLAN_RX_QALLOC	PF Queue Allocation	13.2.2.26.19
0x001D2580	PFLAN_TX_QALLOC	PF Queue Allocation	13.2.2.26.20
0x00280000 + 0x2000*n + 0x4*QRX, n=0...7, QRX=0...2047	QRX_CONTEXT[n,QRX]	Global Receive Queue Context	13.2.2.26.21
0x00290000 + 0x4*QRX, QRX=0...2047	QRX_TAIL[QRX]	Receive Queue Tail Update	13.2.2.26.22
0x00292000 + 0x4*QRX, QRX=0...2047	QRX_ITR[QRX]	Global Receive Queue ITR Expire	13.2.2.26.23
0x002941F8	GLLAN_RCTL_0	Global RLAN Control 0	13.2.2.26.24
0x002941FC	GLLAN_RCTL_1	Global RLAN Control 1	13.2.2.26.25
0x0029420C + 0x4*n, n=0...7	GLLAN_PF_RECIPE[n]	Global PF LAN Recipe	13.2.2.26.26

Table 13-34. PF - LAN Transmit/Receive Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x002D2C00 + 0x4*VP16, VP16=0...15	VPLAN_DSI_VF_MODE[VP16]	VF LAN TXQ Enablement	13.2.2.26.27
0x00440000 + 0x1000*n + 0x4*VSI, n=0...7, VSI=0...767	VSILAN_QTABLE[n,VSI]	VSI Receive Queue Mapping Table	13.2.2.26.28
0x0044c000 + 0x4*VSI, VSI=0...767	VSILAN_QBASE[VSI]	VSI Queue Control	13.2.2.26.29

Table 13-35. PF - TimeSync (IEEE 1588) Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00088808 + 0x4*n, n=0...1	GLTSYN_ENA[n]	Global TimeSync Enable	13.2.2.27.1
0x00088810	GLTSYN_CMD	Global Master TimeSync Command	13.2.2.27.2
0x00088814	GLTSYN_CMD_SYNC	Global Master TimeSync Command SYNC Control	13.2.2.27.3
0x00088818	GLTSYN_SYNC_DLAY	Global TimeSync Sync Delay	13.2.2.27.4
0x0008881C	GLTSYN_HH_DLAY	Global HH Sync Delay	13.2.2.27.5
0x00088880	PFTSYN_SEM	Global TimeSync Semaphore	13.2.2.27.6
0x000888C0 + 0x4*n, n=0...1	GLTSYN_STAT[n]	Global TimeSync Status 0	13.2.2.27.7
0x000888C8 + 0x4*n, n=0...1	GLTSYN_TIME_0[n]	Global TimeSync Time Zero	13.2.2.27.8
0x000888D0 + 0x4*n, n=0...1	GLTSYN_TIME_L[n]	Global TimeSync Time Low	13.2.2.27.9
0x000888D8 + 0x4*n, n=0...1	GLTSYN_TIME_H[n]	Global TimeSync Time High	13.2.2.27.10
0x000888E0 + 0x4*n, n=0...1	GLTSYN_SHTIME_0[n]	Global TimeSync Shadow Time Zero	13.2.2.27.11
0x000888E8 + 0x4*n, n=0...1	GLTSYN_SHTIME_L[n]	Global TimeSync Shadow Time Low	13.2.2.27.12
0x000888F0 + 0x4*n, n=0...1	GLTSYN_SHTIME_H[n]	Global TimeSync Shadow Time High	13.2.2.27.13
0x000888F8 + 0x4*n, n=0...1	GLTSYN_HHTIME_L[n]	Global TimeSync HH Time Low	13.2.2.27.14
0x00088900 + 0x4*n, n=0...1	GLTSYN_HHTIME_H[n]	Global TimeSync HH Time High	13.2.2.27.15
0x00088908 + 0x4*n, n=0...1	GLTSYN_SHADJ_L[n]	Global TimeSync Shadow Adjust Low	13.2.2.27.16
0x00088910 + 0x4*n, n=0...1	GLTSYN_SHADJ_H[n]	Global TimeSync Shadow Adjust High	13.2.2.27.17
0x00088918 + 0x4*n, n=0...1	GLTSYN_INCVAL_L[n]	Global TimeSync Increment Value Low	13.2.2.27.18
0x00088920 + 0x4*n, n=0...1	GLTSYN_INCVAL_H[n]	Global TimeSync Increment Value High	13.2.2.27.19
0x00088928 + 0x4*n, n=0...1	GLTSYN_TGT_L_0[n]	Global TimeSync Target Time Low	13.2.2.27.20
0x00088930 + 0x4*n, n=0...1	GLTSYN_TGT_H_0[n]	Global TimeSync Target Time High	13.2.2.27.21
0x00088938 + 0x4*n, n=0...1	GLTSYN_TGT_L_1[n]	Global TimeSync Target Time Low	13.2.2.27.22
0x00088940 + 0x4*n, n=0...1	GLTSYN_TGT_H_1[n]	Global TimeSync Target Time High	13.2.2.27.23
0x00088948 + 0x4*n, n=0...1	GLTSYN_TGT_L_2[n]	Global TimeSync Target Time Low	13.2.2.27.24
0x00088950 + 0x4*n, n=0...1	GLTSYN_TGT_H_2[n]	Global TimeSync Target Time High	13.2.2.27.25
0x00088958 + 0x4*n, n=0...1	GLTSYN_TGT_L_3[n]	Global TimeSync Target Time Low	13.2.2.27.26
0x00088960 + 0x4*n, n=0...1	GLTSYN_TGT_H_3[n]	Global TimeSync Target Time High	13.2.2.27.27
0x00088968 + 0x4*n, n=0...1	GLTSYN_EVNT_L_0[n]	Global TimeSync Event Time Low	13.2.2.27.28

Table 13-35. PF - TimeSync (IEEE 1588) Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00088970 + 0x4*n, n=0...1	GLTSYN_EVNT_H_0[n]	Global TimeSync Event Time High	13.2.2.27.29
0x00088978 + 0x4*n, n=0...1	GLTSYN_EVNT_L_1[n]	Global TimeSync Event Time Low	13.2.2.27.30
0x00088980 + 0x4*n, n=0...1	GLTSYN_EVNT_H_1[n]	Global TimeSync Event Time High	13.2.2.27.31
0x00088988 + 0x4*n, n=0...1	GLTSYN_EVNT_L_2[n]	Global TimeSync Event Time Low	13.2.2.27.32
0x00088990 + 0x4*n, n=0...1	GLTSYN_EVNT_H_2[n]	Global TimeSync Event Time High	13.2.2.27.33
0x00088998 + 0x4*n, n=0...1	GLTSYN_AUX_OUT_0[n]	Global TimeSync AUX Output Control	13.2.2.27.34
0x000889A0 + 0x4*n, n=0...1	GLTSYN_AUX_OUT_1[n]	Global TimeSync AUX Output Control	13.2.2.27.35
0x000889A8 + 0x4*n, n=0...1	GLTSYN_AUX_OUT_2[n]	Global TimeSync AUX Output Control	13.2.2.27.36
0x000889B0 + 0x4*n, n=0...1	GLTSYN_AUX_OUT_3[n]	Global TimeSync AUX Output Control	13.2.2.27.37
0x000889B8 + 0x4*n, n=0...1	GLTSYN_CLKO_0[n]	Global TimeSync Clock Out Duration	13.2.2.27.38
0x000889C0 + 0x4*n, n=0...1	GLTSYN_CLKO_1[n]	Global TimeSync Clock Out Duration	13.2.2.27.39
0x000889C8 + 0x4*n, n=0...1	GLTSYN_CLKO_2[n]	Global TimeSync Clock Out Duration	13.2.2.27.40
0x000889D0 + 0x4*n, n=0...1	GLTSYN_CLKO_3[n]	Global TimeSync Clock Out Duration	13.2.2.27.41
0x000889D8 + 0x4*n, n=0...1	GLTSYN_AUX_IN_0[n]	Global TimeSync AUX Input Control	13.2.2.27.42
0x000889E0 + 0x4*n, n=0...1	GLTSYN_AUX_IN_1[n]	Global TimeSync AUX Input Control	13.2.2.27.43
0x000889E8 + 0x4*n, n=0...1	GLTSYN_AUX_IN_2[n]	Global TimeSync AUX Input Control	13.2.2.27.44
0x000A41D4	GLHH_ART_CTL	Global Hammock Harbor Timer Control	13.2.2.27.45
0x000A41D8	GLHH_ART_TIME_H	Global Hammock Harbor ART Time High	13.2.2.27.46
0x000A41DC	GLHH_ART_TIME_L	Global Hammock Harbor ART Time Low	13.2.2.27.47
0x000A41E0	GLHH_ART_DATA	Global Hammock Harbor Sync-Start DATA	13.2.2.27.48
0x000A4200	PFHH_SEM	Global Hammock Harbor Semaphore	13.2.2.27.49

Table 13-36. PF - Protocol Engine Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00500000 + 0x4*VF, VF=0...255	VFPE_CQPDB[VF]	Protocol Engine VF CQP Doorbell	13.2.2.28.1
0x00500400 + 0x4*VF, VF=0...255	VFPE_CQPTAIL[VF]	Protocol Engine VF CQP Tail	13.2.2.28.2
0x00500800	PFPE_CQPDB	Protocol Engine CQP Doorbell	13.2.2.28.3
0x00500880	PFPE_CQPTAIL	Protocol Engine CQP Tail	13.2.2.28.4
0x00502000 + 0x4*VF, VF=0...255	VFPE_CQARM[VF]	Protocol Engine VF CQ Arm	13.2.2.28.5
0x00502400 + 0x4*VF, VF=0...255	VFPE_CQACK[VF]	Protocol Engine VF CQ Ack	13.2.2.28.6
0x00502800 + 0x4*VF, VF=0...255	VFPE_AEQALLOC[VF]	Protocol Engine VF AEQ Allocate	13.2.2.28.7
0x00502C00	PFPE_CQARM	Protocol Engine CQ Arm	13.2.2.28.8
0x00502C80	PFPE_CQACK	Protocol Engine CQ Ack	13.2.2.28.9
0x00502D00	PFPE_AEQALLOC	Protocol Engine AEQ Allocate	13.2.2.28.10

Table 13-36. PF - Protocol Engine Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00503000 + 0x4*n, n=0...31	GLPE_VFCQEDROPCNT[n]	Protocol Engine CQE Drop Count	13.2.2.28.11
0x00503080 + 0x4*n, n=0...31	GLPE_VFCEQEDROPCNT[n]	Protocol Engine CEQE Drop Count	13.2.2.28.12
0x00503100 + 0x4*n, n=0...31	GLPE_VFAEQEDROPCNT[n]	Protocol Engine AEQE Drop Count	13.2.2.28.13
0x00503200 + 0x4*n, n=0...7	GLPE_PFCQEDROPCNT[n]	Protocol Engine CQE Drop Count	13.2.2.28.14
0x00503220 + 0x4*n, n=0...7	GLPE_PFCEQEDROPCNT[n]	Protocol Engine CEQE Drop Count	13.2.2.28.15
0x00503240 + 0x4*n, n=0...7	GLPE_PFAEQEDROPCNT[n]	Protocol Engine AEQE Drop Count	13.2.2.28.16
0x00503300	GLPE_CQM_FUNC_INVALIDATE	Protocol Engine CQM Func Invalidate Register	13.2.2.28.17
0x00504000 + 0x4*VF, VF=0...255	VFPE_WQEALLOC[VF]	Protocol Engine VF WQE Allocate Register	13.2.2.28.18
0x00504400	PFPE_WQEALLOC	Protocol Engine WQE Allocate Register	13.2.2.28.19
0x00508000 + 0x4*VF, VF=0...255	VFPE_CCQPSTATUS[VF]	Protocol Engine VF Create CQP Status	13.2.2.28.20
0x00508400 + 0x4*VF, VF=0...255	VFPE_CCQPLOW[VF]	Protocol Engine VF Create CQP Low	13.2.2.28.21
0x00508800 + 0x4*VF, VF=0...255	VFPE_CCQPHIGH[VF]	Protocol Engine VF Create CQP High	13.2.2.28.22
0x00508C00 + 0x4*VF, VF=0...255	VFPE_IPCONFIG0[VF]	Protocol Engine VF IP Config 0	13.2.2.28.23
0x00509000 + 0x4*VF, VF=0...255	VFPE_CQPERRCODES[VF]	Protocol Engine VF CQP Error Codes	13.2.2.28.24
0x00509400 + 0x4*VF, VF=0...255	VFPE_TCPNOWTIMER[VF]	Protocol Engine VF TCP Now Timer	13.2.2.28.25
0x00509800 + 0x4*VF, VF=0...255	VFPE_MRTEIDXMASK[VF]	Protocol Engine VF MRTE Index Mask	13.2.2.28.26
0x0050A000	PFPE_CCQPSTATUS	Protocol Engine Create CQP Status	13.2.2.28.27
0x0050A080	PFPE_CCQPLOW	Protocol Engine Create CQP Low	13.2.2.28.28
0x0050A100	PFPE_CCQPHIGH	Protocol Engine Create CQP High	13.2.2.28.29
0x0050A180	PFPE_IPCONFIG0	Protocol Engine IP Config 0	13.2.2.28.30
0x0050A200	PFPE_CQPERRCODES	Protocol Engine CQP Error Codes	13.2.2.28.31
0x0050A280	PFPE_TCPNOWTIMER	Protocol Engine TCP Now Timer	13.2.2.28.32
0x0050A300	PFPE_MRTEIDXMASK	Protocol Engine MRTE Index Mask	13.2.2.28.33
0x0050B300 + 0x4*n, n=0...31	GLPE_VFTCPNOW50USCNT[n]	Protocol Engine TCP Now 50us Count	13.2.2.28.34
0x0050B400 + 0x4*n, n=0...31	GLPE_VFFLMXMITALLOCERR[n]	Protocol Engine FLM XMIT Allocate Error	13.2.2.28.35
0x0050B480 + 0x4*n, n=0...31	GLPE_VFFLMQ1ALLOCERR[n]	Protocol Engine FLM Q1 Allocate Error	13.2.2.28.36
0x0050B500 + 0x4*n, n=0...31	GLPE_VFFLMRRFALLOCERR[n]	Protocol Engine FLM Read Response Allocate Error	13.2.2.28.37

Table 13-36. PF - Protocol Engine Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0050B580 + 0x4*n, n=0...31	GLPE_VFFLMOOISCALLOCERR[n]	Protocol Engine FLM Out of Order Send Completion (OOISC) Allocate Error	13.2.2.28.38
0x0050B8C0 + 0x4*n, n=0...7	GLPE_PFTCPNOW50USCNT[n]	Protocol Engine TCP Now 50us Count	13.2.2.28.39
0x0050B900 + 0x4*n, n=0...7	GLPE_PFFLMXMITALLOCERR[n]	Protocol Engine FLM XMIT Allocate Error	13.2.2.28.40
0x0050B920 + 0x4*n, n=0...7	GLPE_PFFLMQ1ALLOCERR[n]	Protocol Engine FLM Q1 Allocate Error	13.2.2.28.41
0x0050B940 + 0x4*n, n=0...7	GLPE_PFFLMRRFALLOCERR[n]	Protocol Engine FLM Read Response Allocate Error	13.2.2.28.42
0x0050B960 + 0x4*n, n=0...7	GLPE_PFFLMOOISCALLOCERR[n]	Protocol Engine FLM Out of Order Send Completion (OOISC) Allocate Error	13.2.2.28.43
0x0050BA5C	GLPE_CPUSTATUS0	Protocol Engine CPU Status 0	13.2.2.28.44
0x0050BA60	GLPE_CPUSTATUS1	Protocol Engine CPU Status 1	13.2.2.28.45
0x0050BA64	GLPE_CPUSTATUS2	Protocol Engine CPU Status 2	13.2.2.28.46
0x0050C000	GLPE_PEPM_CTRL	PEPM Control	13.2.2.28.47
0x0050C004	GLPE_PEPM_DEALLOC	PEPM Dealloc	13.2.2.28.48
0x0050C020	GLPE_PEPM_PSQ_COUNT	PEPM PSQ Count	13.2.2.28.49
0x0050C040 + 0x4*n, n=0...511	PRT_PEPM_COUNT[n]	PEPM PSQ/MDQ Count	13.2.2.28.50
0x0050C840 + 0x4*n, n=0...511	GLPE_PEPM_THRESH[n]	PEPM PQ Threshold	13.2.2.28.51
0x0053241C	GLPE_PUSH_PEPM	Push PEPM	13.2.2.28.52
0x00536000 + 0x4*n, n=0...511	GLPE_MDQ_BASE[n]	MDQ Base	13.2.2.28.53
0x00536800 + 0x4*n, n=0...511	GLPE_MDQ_SIZE[n]	MDQ Size	13.2.2.28.54
0x00537000 + 0x4*n, n=0...511	GLPE_MDQ_PTR[n]	MDQ Pointer	13.2.2.28.55

Table 13-37. PF - Manageability Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0008309C	GL_MNG_FW_RAM_STAT	MNG FW RAM Status Registers	13.2.2.29.1
0x00083100	GL_FWRESETCNT	Firmware Reset Count	13.2.2.29.2
0x00083120 + 0x4*n, n=0...7	GL_MNG_SHA_EXTEND[n]	SHA Extend Value	13.2.2.29.3
0x00083148	GL_MNG_SHA_EXTEND_STATUS	SHA Extend Value Status	13.2.2.29.4
0x00083160 + 0x4*n, n=0...7	GL_MNG_SHA_EXTEND_ROM[n]	SHA ROM Extend Value	13.2.2.29.5
0x000B6130	GL_MNG_HWARB_CTRL	Hardware Arbitration Control	13.2.2.29.6
0x000B6134	GL_MNG_FWSM	Firmware Semaphore	13.2.2.29.7
0x000B6180 + 0x4*n, n=0...9	GENERAL_MNG_FW_DBG_CSR[n]	General FW Debug Registers	13.2.2.29.8
0x00214120 + 0x20*n, n=0...3	PRT_MNG_METF[n]	Management Ethernet Type Filters	13.2.2.29.9

Table 13-37. PF - Manageability Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x002141A0 + 0x20*n, n=0...3	PRT_MNG_MIPAF4[n]	Manageability IPv4 Address Filter	13.2.2.29.10
0x00214220 + 0x20*n, n=0...3	PRT_MNG_MMAH[n]	Manageability MAC Address High	13.2.2.29.11
0x002142A0 + 0x20*n, n=0...3	PRT_MNG_MMAL[n]	Manageability MAC Address Low	13.2.2.29.12
0x00214320 + 0x20*n, n=0...15	PRT_MNG_MFUTP[n]	Management Flex UDP/TCP Ports	13.2.2.29.13
0x00214520 + 0x20*n, n=0...15	PRT_MNG_MIPAF6[n]	Manageability IPv6 Address Filter	13.2.2.29.14
0x00214720	PRT_MNG_MANC	Management Control Register	13.2.2.29.15
0x00214740	PRT_MNG_MNGONLY	Management Only Traffic Register	13.2.2.29.16
0x00214760	PRT_MNG_MSFM	Manageability Special Filters Modifiers	13.2.2.29.17
0x00214780 + 0x20*n, n=0...7	PRT_MNG_MAVTV[n]	Management VLAN TAG Value	13.2.2.29.18
0x00214880 + 0x20*n, n=0...7	PRT_MNG_MDEF[n]	Manageability Decision Filters1	13.2.2.29.19
0x00214980 + 0x20*n, n=0...3	PRT_MNG_MDEFVSI[n]	Management Decision Filters VSI	13.2.2.29.20
0x00214A00 + 0x20*n, n=0...7	PRT_MNG_MDEF_EXT[n]	Manageability Decision Filters	13.2.2.29.21
0x00216018 + 0x4*n, n=0...31	GL_SWT_PRT2MDEF[n]	Port to MDEF Set Mapping	13.2.2.29.22

Table 13-38. PF - Malicious Prevention Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00040000 + 0x4*VF, VF=0...255	VP_MDET_TX_TDPU[VF]	Malicious VF Driver Detected on Tx TDPU	13.2.2.30.1
0x00040800	PF_MDET_TX_TDPU	Malicious PF Driver Detected on Tx TDPU	13.2.2.30.2
0x000FB800 + 0x4*VF, VF=0...255	VP_MDET_TX_TCLAN[VF]	Malicious VF Driver Detected on Tx TCLAN	13.2.2.30.3
0x000FC000	PF_MDET_TX_TCLAN	Malicious PF Driver Detected on Tx TCLAN	13.2.2.30.4
0x000FC068	GL_MDET_TX_TCLAN	Malicious Driver Tx Event Details	13.2.2.30.5
0x000FC348 + 0x4*n, n=0...767	VM_MDET_TX_TCLAN[n]	Malicious PF Driver Detected on Tx TCLAN	13.2.2.30.6
0x00294200	GLRLAN_MDET	RLAN Malicious Events	13.2.2.30.7
0x0029422C	GL_MDCK_RX	Malicious Driver Rx Checks Enabled	13.2.2.30.8
0x00294280	PF_MDET_RX	Malicious PF Driver Detected on Rx	13.2.2.30.9
0x00294400 + 0x4*VF, VF=0...255	VP_MDET_RX[VF]	Malicious VF Driver Detected on Rx	13.2.2.30.10
0x00294C00	GL_MDET_RX	Malicious Driver Rx Event Details	13.2.2.30.11
0x002D2000 + 0x4*VF, VF=0...255	VP_MDET_TX_PQM[VF]	Malicious VF Driver Detected on Tx PQM	13.2.2.30.12
0x002D2C80	PF_MDET_TX_PQM	Malicious PF Driver Detected on Tx PQM	13.2.2.30.13
0x002D2DF4	GL_MDCK_CFG1_TX_PQM	Malicious Driver Tx Command Checks PQM Configuration 1	13.2.2.30.14
0x002D2DFC	GL_MDCK_EN_TX_PQM	Malicious Driver Tx Command Checks Enable PQM	13.2.2.30.15
0x002D2E00	GL_MDET_TX_PQM	Malicious Driver Tx Event Details PQM	13.2.2.30.16

Table 13-39. PF - Rx QoS Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00122120	PRTDCB_RRDMAPMS	DCB Receive RDMA Pipe Monitor Status	13.2.2.31.1
0x00122240	PRTDCB_RPPMC	DCB Receive per Port Pipe Monitor Control	13.2.2.31.2
0x00122260	GLDCB_RPCC	DCB Receive Pacing Control	13.2.2.31.3
0x00122280	PRTDCB_RLANPMS	DCB Receive LAN Pipe Monitor Status	13.2.2.31.4
0x001222C0 + 0x4*n, n=0...31	GLDCB_RTCTQ[n]	DCB Receive per TC PFC Timer Queue	13.2.2.31.5
0x00122340 + 0x4*n, n=0...31	GLDCB_RTCTS[n]	DCB Receive per TC PFC Timer Status	13.2.2.31.6
0x001223C0	GLDCB_RSPMS	DCB Receive Shared Pipe Monitor Status	13.2.2.31.7
0x001223C4	GLDCB_RSPMC	DCB Receive Shared Pipe Monitor Control	13.2.2.31.8
0x001223C8	GLDCB_RMPMC	DCB Receive Manageability Pipe Monitor Control	13.2.2.31.9
0x001223CC	GLDCB_RMPMS	DCB Receive Manageability Pipe Monitor Status	13.2.2.31.10
0x001223D0	GLDCB_RTCTI	DCB Receive per TC PFC Timer Indication	13.2.2.31.11
0x001223D4 + 0x4*n, n=0...7	GLRCB_CFG_COTF_CNT[n]	RCB Configuration Change on the Fly Counter	13.2.2.31.12
0x001223F4	GLRCB_CFG_COTF_ST	RCB Configuration Change on the Fly Status	13.2.2.31.13
0x00200308 + 0x4*n, n=0...15	GLRPRS_PMCFG_DPS[n]	Rx PM Dedicated Pool Size	13.2.2.31.14
0x00200388 + 0x4*n, n=0...15	GLRPRS_PMCFG_DHW[n]	Rx PM Dedicated Pool High Watermark	13.2.2.31.15
0x002003C8 + 0x4*n, n=0...15	GLRPRS_PMCFG_DLW[n]	Rx PM Dedicated Pool Low Watermark	13.2.2.31.16
0x00200408 + 0x4*n, n=0...7	GLRPRS_PMCFG_SPS[n]	Rx PM Shared Pool Size	13.2.2.31.17
0x00200448 + 0x4*n, n=0...7	GLRPRS_PMCFG_SHW[n]	Rx PM Shared Pool High Watermark	13.2.2.31.18
0x00200468 + 0x4*n, n=0...7	GLRPRS_PMCFG_SLW[n]	Rx PM Shared Pool Low Watermark	13.2.2.31.19
0x00200488 + 0x4*n, n=0...31	GLRPRS_PMCFG_TC_CFG[n]	TC Pool Config	13.2.2.31.20
0x00200588 + 0x4*n, n=0...31	GLRPRS_PMCFG_TCHW[n]	Rx PM TC High Watermark	13.2.2.31.21
0x00200608 + 0x4*n, n=0...31	GLRPRS_PMCFG_TCLW[n]	Rx PM TC Low Watermark	13.2.2.31.22
0x00204900 + 0x4*n, n=0...31	GLSWT_PMCFG_TC_CFG[n]	TC Pool Config	13.2.2.31.23

Table 13-40. PF - 4 KB Page Mappings

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x02000000 + 0x1000*VSI, VSI=0...767	VSI_MBX_ATQBAL[VSI]	VSI Mailbox Transmit Queue Base Address Low	13.2.2.22.45
0x02000004 + 0x1000*VSI, VSI=0...767	VSI_MBX_ATQBAH[VSI]	VSI Mailbox Transmit Queue Base Address High	13.2.2.22.46
0x02000008 + 0x1000*VSI, VSI=0...767	VSI_MBX_ATQLEN[VSI]	VSI Mailbox Transmit Queue Length	13.2.2.22.47
0x0200000C + 0x1000*VSI, VSI=0...767	VSI_MBX_ATQH[VSI]	VSI Mailbox Transmit Head	13.2.2.22.48
0x02000010 + 0x1000*VSI, VSI=0...767	VSI_MBX_ATQT[VSI]	VSI Mailbox Transmit Tail	13.2.2.22.49
0x02000014 + 0x1000*VSI, VSI=0...767	VSI_MBX_ARQBAL[VSI]	VSI Mailbox Receive Queue Base Address Low	13.2.2.22.50
0x02000018 + 0x1000*VSI, VSI=0...767	VSI_MBX_ARQBAH[VSI]	VSI Mailbox Receive Queue Base Address High	13.2.2.22.51

Table 13-40. PF - 4 KB Page Mappings [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0200001C + 0x1000*VSI, VSI=0...767	VSI_MBX_ARQLEN[VSI]	VSI Mailbox Receive Queue Length	13.2.2.22.52
0x02000020 + 0x1000*VSI, VSI=0...767	VSI_MBX_ARQH[VSI]	VSI Mailbox Receive Head	13.2.2.22.53
0x02000024 + 0x1000*VSI, VSI=0...767	VSI_MBX_ARQT[VSI]	VSI Mailbox Receive Tail	13.2.2.22.54
0x02D00000	PF0_FW_HLP_ATQBAL_PAGE	PF0 HLP Firmware Admin Transmit Queue Base Address Low	13.2.2.22.4
0x02D00010	PF0_MBX_HLP_ATQBAL_PAGE	PF0 HLP Mailbox Transmit Queue Base Address Low	13.2.2.22.115
0x02D00020	PF0_SB_HLP_ATQBAL_PAGE	PF0 HLP Sideband Transmit Queue Base Address Low	13.2.2.22.165
0x02D00080	PF0_FW_HLP_ARQBAL_PAGE	PF0 HLP Firmware Admin Receive Queue Base Address Low	13.2.2.22.8
0x02D00090	PF0_MBX_HLP_ARQBAL_PAGE	PF0 HLP Mailbox Receive Queue Base Address Low	13.2.2.22.120
0x02D000A0	PF0_SB_HLP_ARQBAL_PAGE	PF0 HLP Sideband Receive Queue Base Address Low	13.2.2.22.170
0x02D00100	PF0_FW_HLP_ATQBAH_PAGE	PF0 HLP Firmware Admin Transmit Queue Base Address High	13.2.2.22.12
0x02D00110	PF0_MBX_HLP_ATQBAH_PAGE	PF0 HLP Mailbox Transmit Queue Base Address High	13.2.2.22.116
0x02D00120	PF0_SB_HLP_ATQBAH_PAGE	PF0 HLP Sideband Transmit Queue Base Address High	13.2.2.22.166
0x02D00180	PF0_FW_HLP_ARQBAH_PAGE	PF0 HLP Firmware Admin Receive Queue Base Address High	13.2.2.22.16
0x02D00190	PF0_MBX_HLP_ARQBAH_PAGE	PF0 HLP Mailbox Receive Queue Base Address High	13.2.2.22.121
0x02D001A0	PF0_SB_HLP_ARQBAH_PAGE	PF0 HLP Sideband Receive Queue Base Address High	13.2.2.22.171
0x02D00200	PF0_FW_HLP_ATQLEN_PAGE	PF0 HLP Firmware Admin Transmit Queue Length	13.2.2.22.20
0x02D00210	PF0_MBX_HLP_ATQLEN_PAGE	PF0 HLP Mailbox Transmit Queue Length	13.2.2.22.117
0x02D00220	PF0_SB_HLP_ATQLEN_PAGE	PF0 HLP Sideband Transmit Queue Length	13.2.2.22.167
0x02D00280	PF0_FW_HLP_ARQLEN_PAGE	PF0 HLP Firmware Admin Receive Queue Length	13.2.2.22.24
0x02D00290	PF0_MBX_HLP_ARQLEN_PAGE	PF0 HLP Mailbox Receive Queue Length	13.2.2.22.122
0x02D002A0	PF0_SB_HLP_ARQLEN_PAGE	PF0 HLP Sideband Receive Queue Length	13.2.2.22.172
0x02D00300	PF0_FW_HLP_ATQH_PAGE	PF0 HLP Firmware Admin Transmit Head	13.2.2.22.28
0x02D00310	PF0_MBX_HLP_ATQH_PAGE	PF0 HLP Mailbox Transmit Head	13.2.2.22.118
0x02D00320	PF0_SB_HLP_ATQH_PAGE	PF0 HLP Sideband Transmit Head	13.2.2.22.168
0x02D00380	PF0_FW_HLP_ARQH_PAGE	PF0 HLP Firmware Admin Receive Queue Head	13.2.2.22.32
0x02D00390	PF0_MBX_HLP_ARQH_PAGE	PF0 HLP Mailbox Receive Head	13.2.2.22.123
0x02D003A0	PF0_SB_HLP_ARQH_PAGE	PF0 HLP Sideband Receive Head	13.2.2.22.173
0x02D00400	PF0_FW_HLP_ATQT_PAGE	PF0 HLP Firmware Admin Transmit Tail	13.2.2.22.36

Table 13-40. PF - 4 KB Page Mappings [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x02D00410	PF0_MBX_HLP_ATQT_PAGE	PF0 HLP Mailbox Transmit Tail	13.2.2.22.119
0x02D00420	PF0_SB_HLP_ATQT_PAGE	PF0 HLP Sideband Transmit Tail	13.2.2.22.169
0x02D00480	PF0_FW_HLP_ARQT_PAGE	PF0 HLP Firmware Admin Receive Queue Tail	13.2.2.22.40
0x02D00490	PF0_MBX_HLP_ARQT_PAGE	PF0 HLP Mailbox Receive Tail	13.2.2.22.124
0x02D004A0	PF0_SB_HLP_ARQT_PAGE	PF0 HLP Sideband Receive Tail	13.2.2.22.174
0x02D004B0	GLNVM_AL_DONE_HLP_PAGE	HLP Auto-Load Done Register	13.2.2.7.1
0x02D01000	PF0INT_OICR_HLP_PAGE	PF0 Interrupt Other Cause HLP	13.2.2.15.44
0x02D01100	PF0INT_OICR_ENA_HLP_PAGE	PF0 Interrupt Other Cause HLP Enablement	13.2.2.15.37
0x02D02000	PF0INT_OICR_PSM_PAGE	PF0 Interrupt Other Cause PSM	13.2.2.15.35
0x02D02100	PF0INT_OICR_ENA_PSM_PAGE	PF0 Interrupt Other Cause PSM Enablement	13.2.2.15.40
0x02D03000	PF0INT_OICR_CPM_PAGE	PF0 Interrupt Other Cause CPM	13.2.2.15.34
0x02D03100	PF0INT_OICR_ENA_CPM_PAGE	PF0 Interrupt Other Cause CPM Enablement	13.2.2.15.42
0x02D40000	PF0_FW_PSM_ATQBAL_PAGE	PF0 PSM Firmware Admin Transmit Queue Base Address Low	13.2.2.22.3
0x02D40010	PF0_MBX_PSM_ATQBAL_PAGE	PF0 PSM Mailbox Transmit Queue Base Address Low	13.2.2.22.125
0x02D40080	PF0_FW_PSM_ARQBAL_PAGE	PF0 PSM Firmware Admin Receive Queue Base Address Low	13.2.2.22.7
0x02D40090	PF0_MBX_PSM_ARQBAL_PAGE	PF0 PSM Mailbox Receive Queue Base Address Low	13.2.2.22.130
0x02D40100	PF0_FW_PSM_ATQBAH_PAGE	PF0 PSM Firmware Admin Transmit Queue Base Address High	13.2.2.22.11
0x02D40110	PF0_MBX_PSM_ATQBAH_PAGE	PF0 PSM Mailbox Transmit Queue Base Address High	13.2.2.22.126
0x02D40180	PF0_FW_PSM_ARQBAH_PAGE	PF0 PSM Firmware Admin Receive Queue Base Address High	13.2.2.22.15
0x02D40190	PF0_MBX_PSM_ARQBAH_PAGE	PF0 PSM Mailbox Receive Queue Base Address High	13.2.2.22.131
0x02D40200	PF0_FW_PSM_ATQLEN_PAGE	PF0 PSM Firmware Admin Transmit Queue Length	13.2.2.22.19
0x02D40210	PF0_MBX_PSM_ATQLEN_PAGE	PF0 PSM Mailbox Transmit Queue Length	13.2.2.22.127
0x02D40280	PF0_FW_PSM_ARQLEN_PAGE	PF0 PSM Firmware Admin Receive Queue Length	13.2.2.22.23
0x02D40290	PF0_MBX_PSM_ARQLEN_PAGE	PF0 PSM Mailbox Receive Queue Length	13.2.2.22.132
0x02D40300	PF0_FW_PSM_ATQH_PAGE	PF0 PSM Firmware Admin Transmit Head	13.2.2.22.27
0x02D40310	PF0_MBX_PSM_ATQH_PAGE	PF0 PSM Mailbox Transmit Head	13.2.2.22.128
0x02D40380	PF0_FW_PSM_ARQH_PAGE	PF0 PSM Firmware Admin Receive Queue Head	13.2.2.22.31
0x02D40390	PF0_MBX_PSM_ARQH_PAGE	PF0 PSM Mailbox Receive Head	13.2.2.22.133
0x02D40400	PF0_FW_PSM_ATQT_PAGE	PF0 PSM Firmware Admin Transmit Tail	13.2.2.22.35

Table 13-40. PF - 4 KB Page Mappings [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x02D40410	PF0_MBX_PSM_ATQT_PAGE	PF0 PSM Mailbox Transmit Tail	13.2.2.22.129
0x02D40480	PF0_FW_PSM_ARQT_PAGE	PF0 PSM Firmware Admin Receive Queue Tail	13.2.2.22.39
0x02D40490	PF0_MBX_PSM_ARQT_PAGE	PF0 PSM Mailbox Receive Tail	13.2.2.22.134
0x02D80010	PF0_MBX_CPM_ATQBAL_PAGE	PF0 CPM Mailbox Transmit Queue Base Address Low	13.2.2.22.105
0x02D80020	PF0_SB_CPM_ATQBAL_PAGE	PF0 CPM Sideband Transmit Queue Base Address Low	13.2.2.22.135
0x02D80090	PF0_MBX_CPM_ARQBAL_PAGE	PF0 CPM Mailbox Receive Queue Base Address Low	13.2.2.22.110
0x02D800A0	PF0_SB_CPM_ARQBAL_PAGE	PF0 CPM Sideband Receive Queue Base Address Low	13.2.2.22.140
0x02D80110	PF0_MBX_CPM_ATQBAH_PAGE	PF0 CPM Mailbox Transmit Queue Base Address High	13.2.2.22.106
0x02D80120	PF0_SB_CPM_ATQBAH_PAGE	PF0 CPM Sideband Transmit Queue Base Address High	13.2.2.22.136
0x02D80190	PF0_MBX_CPM_ARQBAH_PAGE	PF0 CPM Mailbox Receive Queue Base Address High	13.2.2.22.111
0x02D801A0	PF0_SB_CPM_ARQBAH_PAGE	PF0 CPM Sideband Receive Queue Base Address High	13.2.2.22.141
0x02D80210	PF0_MBX_CPM_ATQLEN_PAGE	PF0 CPM Mailbox Transmit Queue Length	13.2.2.22.107
0x02D80220	PF0_SB_CPM_ATQLEN_PAGE	PF0 CPM Sideband Transmit Queue Length	13.2.2.22.137
0x02D80290	PF0_MBX_CPM_ARQLEN_PAGE	PF0 CPM Mailbox Receive Queue Length	13.2.2.22.112
0x02D802A0	PF0_SB_CPM_ARQLEN_PAGE	PF0 CPM Sideband Receive Queue Length	13.2.2.22.142
0x02D80310	PF0_MBX_CPM_ATQH_PAGE	PF0 CPM Mailbox Transmit Head	13.2.2.22.108
0x02D80320	PF0_SB_CPM_ATQH_PAGE	PF0 CPM Sideband Transmit Head	13.2.2.22.138
0x02D80390	PF0_MBX_CPM_ARQH_PAGE	PF0 CPM Mailbox Receive Head	13.2.2.22.113
0x02D803A0	PF0_SB_CPM_ARQH_PAGE	PF0 CPM Sideband Receive Head	13.2.2.22.143
0x02D80410	PF0_MBX_CPM_ATQT_PAGE	PF0 CPM Mailbox Transmit Tail	13.2.2.22.109
0x02D80420	PF0_SB_CPM_ATQT_PAGE	PF0 CPM Sideband Transmit Tail	13.2.2.22.139
0x02D80490	PF0_MBX_CPM_ARQT_PAGE	PF0 CPM Mailbox Receive Tail	13.2.2.22.114
0x02D804A0	PF0_SB_CPM_ARQT_PAGE	PF0 CPM Sideband Receive Tail	13.2.2.22.144
0x02F00000 + 0x1000*DBLQ, DBLQ=0...255	QTX_COMM_DBLQ_DBELL_PAGE[DBLQ]	Transmit Comm Scheduler Queue Doorbell	13.2.2.25.29
0x03000000 + 0x1000*n, n=0...2047	PF0INT_DYN_CTL[n]	PF0 Interrupt Dynamic Control	13.2.2.15.10
0x03000004 + 0x1000*n, n=0...2047	PF0INT_ITR_0[n]	PF Interrupt Throttling 0	13.2.2.15.7
0x03000008 + 0x1000*n, n=0...2047	PF0INT_ITR_1[n]	PF Interrupt Throttling 1	13.2.2.15.7
0x0300000C + 0x1000*n, n=0...2047	PF0INT_ITR_2[n]	PF Interrupt Throttling 2	13.2.2.15.7

Table 13-40. PF - 4 KB Page Mappings [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x03800000 + 0x1000*QRX, QRX=0...2047	QRX_TAIL_PAGE[QRX]	Receive Queue Tail Update	13.2.2.26.22
0x04000000 + 0x1000*DBQM, DBQM=0...16383	QTX_COMM_DBELL_PAGE[DBQM]	Transmit Comm Scheduler Queue Doorbell	13.2.2.25.27

13.2.2 Detailed Register Descriptions - PF BAR0

13.2.2.1 PF - General Registers

This category contains registers for general device control and status.

13.2.2.1.1 VF Reset Status - VFGEN_RSTAT[VF] (0x00074000 + 0x4*VF, VF=0...255; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VFR_STATE	1:0	00b	RW	UNDEFINED	**VFR State** Defines the VFR reset progress as follows: 00b = VFR in progress. 01b = VFR completed. All other values are reserved. This field is used to communicate the reset progress to the VF with no impact on hardware functionality.
RESERVED	31:2	0x0	RSV	N/A	Reserved.

13.2.2.1.2 Firmware Status Register - GL_FWSTS (0x00083048; RO)

This register reports the status of the EMP.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FWS0B	7:0	0x0	RO	N/A	**Firmware Status 0 Byte** This byte is RO through the host interface.
FWROWD	8	0b	RW1C	DYNAMIC	**Firmware Reset on Watchdog Indication** Set when a firmware reset is asserted due to watchdog expiration. Cleared when the host writes a 1b to it. Writing a 0b to this bit does not change its value. ***Note:*** This bit is also set after LAN_PWR_GOOD and is RO through the AUX interface.
FWRI	9	1b	RW1C	DYNAMIC	**Firmware Reset Indication** Set when a firmware reset is asserted. Cleared when the host writes a 1b to it. Writing a 0b to this bit does not change its value. ***Note:*** This bit is also set after LAN_PWR_GOOD and is RO through the AUX interface.
RESERVED	15:10	0x0	RSV	N/A	Reserved.
FWS1B	23:16	0x0	RO	N/A	**Firmware Status 1 Byte** This byte is RO through the host interface.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.1.3 PF State - PFGEN_STATE (0x00088000; RW)

This register defines to software the main characteristics of the PF. It does not have any direct impact on device functionality.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PFPEEN	0	0b	RW	UNDEFINED	**PF PE Enable** 0b = Protocol engine services should not be used for this function. 1b = Protocol engine services can be used for this function.
RESERVED	1	0b	RSV	N/A	Reserved.
PFLINKEN	2	0b	RW	UNDEFINED	**PF Link Enable** 0b = The PF driver is disabled. The software device driver must at minimum report it does not have link. 1b = The PF driver is enabled.
PFSCEN	3	0b	RW	UNDEFINED	**PF iSCSI Enable** 0b = iSCSI should not be used for this function. 1b = iSCSI can be used for this function.
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.1.4 Global GPIO Control - GLGEN_GPIO_CTL[n] (0x000880C8 + 0x4*n, n=0...6; RW)

These registers control the mode of operation of the following I/O signals:

- Registers 0...3 control SDP_TIMESYNC[0:3], respectively.
- Registers 4 controls the TIME_SYNC (in) signal.
- Registers 5 controls the 1PPS (out) signal.
- Register 6 control the CLK_SYNCE (out) signal.

This register is initialized only at LAN Power Good, preserving the GPIO states across software and PCIe resets.

Field	Bit(s)	Init.	Type	CFG Policy	Description
IN_VALUE	0	0b	RO	N/A	**In Value** Reflect the state of the GPIO (valid for both cases: input and output signal).
IN_TRANSIT	1	0b	RCW	UNDEFINED	**In Transition** This bit is set to one following any transition on the GPIO. Writing "1" to this bit clears it.
OUT_VALUE	2	0b	RW	UNDEFINED	**Out Value** Defines the output level driven on this GPIO. It is meaningful only when the *PIN_DIR* is configured as output.
NO_P_UP	3	1b	RW	UNDEFINED	**No Pull-Up** Defines if GPIO pin has an internal pull-up. 0b = Pull-up 1b = No pull-up
PIN_DIR	4	0b	RW	UNDEFINED	**Pin Direction** Controls whether this GPIOn pin is configured as an input or output. 0b = Input 1b = Output

Field	Bit(s)	Init.	Type	CFG Policy	Description
TRI_CTL	5	0b	RW	UNDEFINED	**Tristate Control** Meaningful only for the single-ended GPIOs when configured as output and driven to high. 0b = The pin is tri-stated. 1b = The pin is driven to high.
RESERVED	7:6	00b	RSV	N/A	Reserved.
PIN_FUNC	11:8	0x0	RW	UNDEFINED	**Pin Function** This field controls the functionality of this GPIOn pin. 0x0 = SDP (software definable input or output). 0x1 = Input Event sampled by EVENT register 0 and AUX_IN 0 of master timer 0. 0x2 = Input Event sampled by EVENT register 1 and AUX_IN 1 of master timer 0. 0x3 = Input Event sampled by EVENT register 2 and AUX_IN 2 of master timer 0. 0x4 = Input Event sampled by EVENT register 0 and AUX_IN 0 of master timer 1. 0x5 = Input Event sampled by EVENT register 1 and AUX_IN 1 of master timer 1. 0x6 = Input Event sampled by EVENT register 2 and AUX_IN 2 of master timer 1. 0x8 = Output signal initiated by TGT register 0 and AUX_OUT 0 of master timer 0. 0x9 = Output signal initiated by TGT register 1 and AUX_OUT 1 of master timer 0. 0xA = Output signal initiated by TGT register 2 and AUX_OUT 2 of master timer 0. 0xB = Output signal initiated by TGT register 3 and AUX_OUT 3 of master timer 0. 0xC = Output signal initiated by TGT register 0 and AUX_OUT 0 of master timer 1. 0xD = Output signal initiated by TGT register 1 and AUX_OUT 1 of master timer 1. 0xE = Output signal initiated by TGT register 2 and AUX_OUT 2 of master timer 1. 0xF = Output signal initiated by TGT register 3 and AUX_OUT 3 of master timer 1.
INT_MODE	13:12	00b	RW	UNDEFINED	**Interrupt Mode** This field selects the interrupt mode for this GPIOn. 00b = No interrupt. 01b = Interrupt on rising edge. 10b = Interrupt on falling edge. 11b = Interrupt on any transition.
RESERVED	31:14	0x0	RSV	N/A	Reserved.

13.2.2.1.5 VF Reset Trigger - VPGEN_VFRTRIG[VF] (0x00090000 + 0x4*VF, VF=0...255; RW)

This register affects the VF but exposed only to the parent PF.

Field	Bit(s)	Init.	Type	CFG Policy	Description
VFSWR	0	0b	RW	UNDEFINED	**VF Software Reset** VF software reset is done by the PF setting the *VFSWR* bit. At reset completion, the PF clears this bit.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.1.6 VF Reset Status - VPGEN_VFRSTAT[VF] (0x00090800 + 0x4*VF, VF=0...255; RO)

This register affects the VF but exposed only to the parent PF.

Field	Bit(s)	Init.	Type	CFG Policy	Description
VFRD	0	1b	RO	N/A	**VF Reset Done** VF Software Reset Done indication. This flag is cleared when the *VFSWR* bit is set in the VPGEN_VFRTRIG register. It is set back to 1b when the hardware completes its hardware data path cleanup for the VF, or the *VFSWR* bit is cleared in the VPGEN_VFRTRIG register.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.1.7 PFGEN Control - PFGEN_CTRL (0x00091000; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PFSWR	0	0b	RO	UNDEFINED	**PF Software Reset** Set on reset event from IOSF block and when software sets the bit. Cleared only by firmware writing 0 to it (also cleared on core reset).
RESERVED	31:1	0x0	RSV	N/A	Reserved.

Note: PF reset.

13.2.2.1.8 PFR STAT - PFGEN_PFRSTAT (0x00091080; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PFRD	0	1b	RO	UNDEFINED	**PF Reset Done** *PFRD* is cleared when GL_XLR_MARKER_TRIG _VMLR sets the relevant PF reset. Set when hardware cleanup is done.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

Note: PF reset.

13.2.2.1.9 PF Driver Unload - PFGEN_DRUN (0x00091180; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DRVUNLD	0	0b	RW	UNDEFINED	**Driver Unload** The PF sets this bit to indicate its driver is unloading.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.1.10 VM Reset Trigger - VSIGEN_RTRIG[VSI] (0x00091800 + 0x4*VSI, VSI=0...767; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VMSWR	0	0b	RW	UNDEFINED	**VM Software Reset** Initiated by setting the *VMR* bit. At completion of the reset flow, the PF software clears this bit.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.1.11 VM Reset Status - VSIGEN_RSTAT[VSI] (0x00092800 + 0x4*VSI, VSI=0...767; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VMRD	0	1b	RO	N/A	**VM Reset Done** Cleared when *VMSWR* bit is set in the VSIGEN_VMRTRIG register. Set when hardware cleanup for the VM is completed or when the *VMSWR* bit is cleared in the VSIGEN_VMRTRIG register.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.1.12 XLR Marker Trigger - GL_XLR_MARKER_TRIG_VMLR (0x00093804; RW)

Field definitions are the same as those defined in Section 13.2.2.10.5.

13.2.2.1.13 Global Marker Count - GLGEN_MARKER_COUNT (0x000939E8; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MARKER_COUNT	7:0	0x0	RW	UNDEFINED	**Marker Count** Specifies the value of markers to count for specific client to decide on XLR completion. Valid when *MARKER_COUNT_EN* is set.
RESERVED	30:8	0x0	RSV	N/A	Reserved.
MARKER_COUNT_EN	31	0b	RW	UNDEFINED	**Marker Count Enable** Specifies if the client marker count value is valid and should override the one chosen from the system state configured.

13.2.2.1.14 Global Wait Between Transaction Count - GLGEN_XLR_TRNS_WAIT_COUNT (0x000939EC; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
W_BTWN_TRNS_COUNT	4:0	0x8	RW	UNDEFINED	**Wait Between Transaction Count** Number of cycles to wait between two consequent transactions.
RESERVED	7:5	000b	RSV	N/A	Reserved.
W_PEND_TRNS_COUNT	15:8	0x20	RW	UNDEFINED	**Wait Pending Transactions Count** Number of cycles to wait between two pending transactions.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.1.15 Global Wait for HLP After CORER - GLGEN_XLR_MSK2HLP_RDY (0x000939F0; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GLGEN_XLR_MSK2HLP_RDY	0	1b	RW	UNDEFINED	**XLR Mask 2 HLP Ready** Bit should be set on core reset and can be cleared by software writing into it.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.1.16 ECC Error Mask Low - GLGEN_ECC_ERR_RST_MASK_L (0x000939F4; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CLIENT_NUM	31:0	0xFFFFFFFF	RW	N/A	**Client Number** Defines on which source blocks the reset indication is set. Each bit represents masking to a different client according to the following mapping: MSB: mbx_unc_ecc_err, tclan_unc_ecc_err, pqm_unc_ecc_err, psm_unc_ecc_err, fpmat_unc_ecc_err, stat_mac_ecc_int, stat_tdpu_ecc_int, pe_unc_ecc_err, foc_unc_ecc_err, int_unc_ecc_err, rlan_unc_ecc_err, tdpu_unc_ecc_err, rdpu_unc_ecc_err, tcb_unc_ecc_err, rcb_unc_ecc_err, rcupst_unc_ecc_err, rcuswr_unc_ecc_err, rcurprs_unc_ecc_err, rcutprs_unc_ecc_err, rcukgg_unc_ecc_err, rcukggext_unc_ecc_err, rcuaclext_unc_ecc_err, rcupstext_unc_ecc_err, tpb_unc_ecc_err, mac_unc_ecc_err, rpb_unc_ecc_err, pprs_unc_ecc_err_0, pprs_unc_ecc_err_1, pprs_unc_ecc_err_2, pprs_unc_ecc_err_3, pcie_unc_ecc_err, LSB: mng_unc_ecc_err

13.2.2.1.17 ECC Error Mask High - GLGEN_ECC_ERR_RST_MASK_H (0x000939F8; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CLIENT_NUM	6:0	0x7F	RW	N/A	**Client Number** Defines on which source blocks the reset indication is set. Each bit represents masking to a different client according to the following mapping: MSB: stat_rcuacl_ecc_int, stat_rcuswr_ecc_int, stat_rcupst_ecc_int, rcuacl_unc_ecc_err, fds_ecc_int, csr_ecc_int, LSB: tcvmlr_unc_ecc_err
RESERVED	31:7	0x0	RSV	N/A	Reserved.

13.2.2.1.18 ECC Error Int Mask Low - GLGEN_ECC_ERR_INT_TOG_MASK_L (0x000939FC; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CLIENT_NUM	31:0	0xFFFFFFFF	RW	N/A	**Client Number** Defines on which source blocks the interrupt indication is toggled. Each bit represents masking to a different client according to the following mapping: MSB: mbx_unc_ecc_err, tclan_unc_ecc_err, pqm_unc_ecc_err, psm_unc_ecc_err, fpmat_unc_ecc_err, stat_mac_ecc_int, stat_tdpu_ecc_int, pe_unc_ecc_err, foc_unc_ecc_err, int_unc_ecc_err, rlan_unc_ecc_err, tdpu_unc_ecc_err, rdpu_unc_ecc_err, tcb_unc_ecc_err, rcb_unc_ecc_err, rcupst_unc_ecc_err, rcuswr_unc_ecc_err, rcurprs_unc_ecc_err, rcutprs_unc_ecc_err, rcukgg_unc_ecc_err, rcukggext_unc_ecc_err, rcuaclext_unc_ecc_err, rcupstext_unc_ecc_err, tpb_unc_ecc_err, mac_unc_ecc_err, rpb_unc_ecc_err, pprs_unc_ecc_err_0, pprs_unc_ecc_err_1, pprs_unc_ecc_err_2, pprs_unc_ecc_err_3, pcie_unc_ecc_err, LSB: mng_unc_ecc_err

13.2.2.1.19 ECC Error Int Mask High - GLGEN_ECC_ERR_INT_TOG_MASK_H (0x00093A00; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CLIENT_NUM	6:0	0x7F	RW	N/A	**Client Number** Defines on which source blocks the interrupt indication is toggled. Each bit represents masking to a different client according to the following mapping: MSB: stat_rcuacl_ecc_int, stat_rcuswr_ecc_int, stat_rcupst_ecc_int, rcuacl_unc_ecc_err, fds_ecc_int, csr_ecc_int, LSB: tcvmlr_unc_ecc_err
RESERVED	31:7	0x0	RSV	N/A	Reserved.

13.2.2.1.20 PFR STAT - GLGEN_VFLRSTAT[n] (0x00093A04 + 0x4*n, n=0...7; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VFLRS	31:0	0x0	RW1C	UNDEFINED	**VFLRS Flags** PFRD is cleaned when GL_XLR_MARKER_TRIG _VMLR set the relevant PF reset. Cleared when hardware cleanup is done.

Note: PF reset.

13.2.2.1.21 TCVMLR XLR Marker Trigger - GL_XLR_MARKER_TRIG_TCVMLR (0x000A2000; RO)

This register is used to initiate XLR marker reset flow. When the register is written, TCVMLR starts drain marker flow.

Field definitions are the same as those defined in Section 13.2.2.10.5.

13.2.2.1.22 Transmit Scheduler Queue Control - GL_TCVMLR_QCTL (0x000A2004; RO)

This register is used to initiate queue halt/marker flow. Flow state can be polled using REQ_STAT and STAT registers.

Field	Bit(s)	Init.	Type	CFG Policy	Description
QID	13:0	0x0	RW	UNDEFINED	**Queue ID** Queue ID to perform action on: 0-16383
OP	14	0b	RW	UNDEFINED	**Operation** 0b = Halt only. Queue halt request is made to PQM. 1b = Marker only. Only a TC drain marker is sent for the queue.
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.1.23 TCVMLR Drain Marker Control - GL_TCVMLR_DRAIN_MARKER (0x000A2008; RO)

This register is used to initiate a TC drain marker to be sent to the pipe.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PORT	2:0	000b	RW	UNDEFINED	**Port** Port number to send marker to.
TC	7:3	0x0	RW	UNDEFINED	**TC** TC number to send marker to.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.1.24 TCVMLR Halt Done Down Counter - GL_TCVMLR_QCNTR (0x000A200C; RO)

This register is used to set the number of queue halts that need to complete their flow before hardware initiates interrupt.

Field	Bit(s)	Init.	Type	CFG Policy	Description
CNTR	14:0	0x0	RW	UNDEFINED	**Counter** Number of queue halts that need to complete their flow before hardware initiates interrupt. Value is decremented after each halt done, and interrupt is initiated when value reaches 0. If value is already 0 on halt done, no interrupt is initiated.
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.1.25 TCVMLR Queue Port TC Config Control - GL_TCVMLR_QCFG (0x000A2010; RO)

This register is used to configure or request to read port and TC number per queue. They must be configured before hardware can operate correctly.

Field	Bit(s)	Init.	Type	CFG Policy	Description
QID	13:0	0x0	RW	UNDEFINED	**Queue ID**
OP	14	0b	RW	UNDEFINED	**Operation** 0b = Read configuration. Only QID needs to be valid. 1b = Write configuration. All fields must be valid.
PORT	17:15	000b	RW	UNDEFINED	**Port** Port associated with the queue. Valid only on write operation.
TC	22:18	0x0	RW	UNDEFINED	**TC** TC associated with the queue. Valid only on write operation.
RESERVED	31:23	0x0	RSV	N/A	Reserved.

13.2.2.1.26 TCVMLR Queue Port TC Config Status - GL_TCVMLR_QCFG_RD (0x000A2014; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
QID	13:0	0x0	RO	N/A	**Queue ID**
PORT	16:14	000b	RO	N/A	**Port** Port associated with the queue.
TC	21:17	0x0	RO	N/A	**TC** TC associated with the queue.
RESERVED	31:22	0x0	RSV	N/A	Reserved.

13.2.2.1.27 TCVMLR Req Flow Status Control - GL_TCVMLR_REQ_STAT (0x000A2018; RO)

This register is used to request or set status of the different disable/reset flows.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ENT_TYPE	2:0	000b	RW	UNDEFINED	**Entity Type** 000b = VF 001b = VM 010b = PF 011b = Queue All other values are reserved.
ENT_ID	16:3	0x0	RW	UNDEFINED	**Entity ID** • VF/VM/PF = Function number • Queue = Queue ID
OP	17	0b	RW	UNDEFINED	**Operation** 0b = Read status. *WRITE_STATUS* does not need to be valid. 1b = Write status.
WRITE_STATUS	20:18	000b	RW	UNDEFINED	**Write Status** Status to be written (as specified in the GL_TCVMLR_STAT register).
RESERVED	31:21	0x0	RSV	N/A	Reserved.

13.2.2.1.28 TCVMLR Req Flow Status Read - GL_TCVMLR_STAT (0x000A201C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
ENT_TYPE	2:0	000b	RO	N/A	**Entity Type** 000b = VF 001b = VM 010b = PF 011b = Queue All other values are reserved.
ENT_ID	16:3	0x0	RO	N/A	**Entity ID** • VF/VM/PF = Function number • Queue = Queue ID

Field	Bit(s)	Init.	Type	CFG Policy	Description
STATUS	19:17	000b	RO	N/A	**Status** Status to write for specified reset entity. Reset status: 000b = Normal. No request is pending. 001b = Halt scheduling request pending. Firmware triggered queue halt request and flow is yet served by hardware (relevant only for Q entity). 010b = Drain marker request pending. Firmware triggered drain marker request and flow is yet served by hardware. 011b = Done: • For Q marker flow status is set to done when queues markers exit the pipe (pipe is drained). • For Q halt flow status is set to done when halt request was completed. • For VF/VM/PF flow status is set to done when marker request exits TCVMLR block.
RESERVED	31:20	0x0	RSV	N/A	Reserved.

13.2.2.1.29 TCVMLR Req Flow Error Status - GL_TCVMLR_ERR_STAT (0x000A2024; RO)

Error indication register. Saves data for last error made by firmware (initiating new flow for request already in progress).

Field	Bit(s)	Init.	Type	CFG Policy	Description
ERROR	0	0b	RW	UNDEFINED	**Error** When high, indicates a firmware error was made. Firmware error is when a new reset flow is initiated for a reset in progress.
FW_REQ	1	0b	RW	UNDEFINED	**Firmware Request** Indicates what type of firmware request caused the error. 0b = Firmware tried to initiate queue halt flow. 1b = Firmware tried to initiate marker flow.
STAT	4:2	000b	RW	UNDEFINED	**Status** Reset status at time of firmware request. 000b = Normal. No request is pending. 001b = Halt scheduling request pending. Firmware triggered queue halt request and flow is yet served by hardware (relevant only for Q entity). 010b = Drain marker request pending. Firmware triggered drain marker request and flow is yet served by hardware. 011b = Done. Flow triggered by firmware is done and no request is pending (indicates Q halt done or marker done according to the last flow that was initiated).
ENT_TYPE	7:5	000b	RW	UNDEFINED	**Entity Type** 000b = VF 001b = VM 010b = PF 011b = Queue 100b = TC drain marker All other values are reserved.
ENT_ID	21:8	0x0	RW	UNDEFINED	**Entity ID** • VF/VM/PF = Function number • Queue = Queue ID • TC drain marker = 3b Port number follow by 5b TC number: {PORT, TC}

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	31:22	0x0	RSV	N/A	Reserved.

13.2.2.1.30 TCVMLR Drain Done Count for TCLAN - GL_TCVMLR_DRAIN_DONE_TCLAN[n] (0x000A20A8 + 0x4*n, n=0...31; RO)

Register per TC that shows amount of drain marker requests that are done on the TCLAN interface. Lowest instance address corresponds to TC 0, highest to TC 31.

Field	Bit(s)	Init.	Type	CFG Policy	Description
COUNT	7:0	0x0	RO	N/A	**Count** Number of drain markers done for a specific TC on TCLAN interface. Can decrement value using GL_TCVMLR_DRAIN_DONE_DEC.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.1.31 TCVMLR Drain Done Count for TPB - GL_TCVMLR_DRAIN_DONE_TPB[n] (0x000A2128 + 0x4*n, n=0...31; RO)

Register per TC that shows amount of drain marker requests that are done on the TPB interface. Lowest instance address corresponds to TC 0, highest to TC 31.

Field	Bit(s)	Init.	Type	CFG Policy	Description
COUNT	7:0	0x0	RO	N/A	**Count** Number of drain markers done for a specific TC on TPB interface. Can decrement value using GL_TCVMLR_DRAIN_DONE_DEC.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.1.32 TCVMLR Drain Done Decrement Control - GL_TCVMLR_DRAIN_DONE_DEC (0x000A21A8; RO)

This register is used to decrement the per-TC drain done registers.

Field	Bit(s)	Init.	Type	CFG Policy	Description
TARGET	0	0b	RW	UNDEFINED	**Target** 0b = TCLAN register. 1b = TPB register.
INDEX	5:1	0x0	RW	UNDEFINED	**Index** TC index to decrement from (0-31). Value needs to be calculated based on port/TC number and MAC topology.
VALUE	13:6	0x0	RW	UNDEFINED	**Value** Value to decrement from counter. Needs to be less than or equal to counter value to work correctly.
RESERVED	31:14	0x0	RSV	N/A	Reserved.

13.2.2.1.33 TCVMLR Drain Done Down Counter - PRT_TCVMLR_DRAIN_CNTR (0x000A21C0; RO)

Status register showing the number of TC drain markers per port that need to complete their flow before hardware initiates interrupt.

Field	Bit(s)	Init.	Type	CFG Policy	Description
CNTR	13:0	0x0	RO	N/A	**Counter** Number of TC drain markers that need to complete their flow time 2 (x2) before hardware initiates interrupt. Value is decremented after each marker done from both TPB and TCLAN interfaces. An Interrupt is initiated when value reaches 0. If value is already 0 on marker done, no interrupt is initiated.
RESERVED	31:14	0x0	RSV	N/A	Reserved.

13.2.2.1.34 TCVMLR Drain Done Down Counter Control - GL_TCVMLR_DRAIN_CNTR_CTL (0x000A21E0; RO)

This register is used to increment or decrement the per-port TC drain done down counters, which generate an interrupt upon reaching zero.

Field	Bit(s)	Init.	Type	CFG Policy	Description
OP	0	0b	RW	UNDEFINED	**Operation** 0b = Decrement value of counter of specified port. 1b = Increment value of counter of specified port.
PORT	3:1	000b	RW	UNDEFINED	**Port** Port number of the counter to increment/decrement.
VALUE	17:4	0x0	RW	UNDEFINED	**Value** Number of TC drain markers that need to complete their flow time 2 (x2) before hardware initiates interrupt. This value is incremented/decremented from counter.
RESERVED	31:18	0x0	RSV	N/A	Reserved.

13.2.2.1.35 Global Status - GLGEN_STAT (0x000B612C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	31:0	0x0	RSV	N/A	Reserved.

13.2.2.1.36 General Port Status - PRTGEN_STATUS (0x000B8100; RO)

This register contains general port status information.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PORT_VALID	0	0b	RO	N/A	**Port Valid** Denotes if the respective Ethernet port is enabled. 0b = Port is disabled. 1b = Port is enabled.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PORT_ACTIVE	1	0b	RO	N/A	**Port Active** Denotes if the respective Ethernet port is active. 0b = Port is inactive and its respective PHY is in power down. 1b = Port is active and its respective PHY is powered up. PORT_ACTIVE is 1b when: PORT_VALID = 1b AND If (device is in D0 state) then port should be active: If the software device driver activated the link (ACTIVATE_PORT_LINK = 1b) or link is used for manageability. Else, when device is in Dr state, the port should be active+ If link is used for manageability or WoL.
RESERVED	31:2	0x0	RSV	N/A	Reserved.

13.2.2.1.37 General Port Configuration - PRTGEN_CNF (0x000B8120; RO)

This register contains configuration per Ethernet port loaded from NVM.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PORT_DIS	0	1b	RW	UNDEFINED	**Port Disable** Defines if the Ethernet port is enabled from the NVM. 0b = Enabled 1b = Disabled Exception: Port 0 is always enabled and cannot be disabled from the NVM.
ALLOW_PORT_DIS	1	0b	RW	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	RW	UNDEFINED	**EMP Port Disable** 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	N/A	Reserved.

13.2.2.1.38 General Port Configuration2 - PRTGEN_CNF2 (0x000B8160; RO)

This register contains configuration per Ethernet port loaded from NVM.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ACTIVATE_PORT_LINK	0	0b	RW	UNDEFINED	**Activate Port Link** When this field is set to 0b, the port's link is powered down. This field can be used by an application to disable the link until the software device driver is loaded and enables the link. ***Notes:*** 1. The PCIe functions associated with the port are not affected by the link loss. 2. Deactivating the link, using this configuration, is ignored when the interface is used for manageability. To implement this, hardware masks this configuration when the relevant port's PRTPM_GC.*EMP_LINK_ON* is set to 1b.
RESERVED	31:2	0x0	RSV	N/A	Reserved.

13.2.2.1.39 Global Reset Control - GLGEN_RSTCTL (0x000B8180; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GRSTDEL	5:0	0x8	RW	UNDEFINED	**Global Reset Delay** Global/Core and EMP resets delay defined in 100 ms units. Setting *GRSTDEL* to zero bypasses the delay counter.
RESERVED	7:6	00b	RSV	N/A	Reserved.
ECC_RST_ENA	8	0b	RW	UNDEFINED	**ECC Reset Enable** Graceful CORER reset on ECC in any memory other than the EMP memories. 0b = A detected ECC error on these memories generates only an interrupt to the PFs. 1b = A detected ECC error generated a graceful CORER. An ECC_ERR reset event is carried out by firmware. CAR creates an interrupt to firmware with reset type CORER. Firmware triggers reset using the immediate reset triggers.
RESERVED	29:9	0x0	RSV	N/A	Reserved.
ECC_RT_EN	30	0b	RW	UNDEFINED	**ECC Reset Enable** ECC reset flow. Set to 1b to enable legacy flow.
FLR_RT_EN	31	0b	RW	UNDEFINED	**FLR Reset Enable** FLR reset flow. Set to 1b to enable legacy flow.

13.2.2.1.40 Global Clock Status - GLGEN_CLKSTAT (0x000B8184; RO)

This register holds internal clock speeds.

Field	Bit(s)	Init.	Type	CFG Policy	Description
U_CLK_SPEED	2:0	000b	RO	N/A	**Upper Clock Speed** Debug feature. Represents the current speed of the upper clock (core_clk) as follows: 00b = 390.625 MHz. 01b = 195.3125 MHz. 10b = 97.65625 MHz 11b = Reserved
L_CLK_SPEED	5:3	000b	RO	N/A	**Lower Clock Speed** Debug feature. Represents the current speed of the Rx clock for MAC 0. Speeds are represented as follows: Three bits per port (synchronized to the equivalent MAC clock): 000b = 100 Mb/s 001b = 1 GbE 010b = 10 GbE 011b = 40 GbE All other values are reserved.
PSM_CLK_SPEED	8:6	000b	RO	N/A	**PSM Clock Speed** Debug feature. Represents the current speed of the Rx clock for MAC 1. Speeds are represented as follows: Three bits per port (synchronized to the equivalent MAC clock): 000b = 100 Mb/s 001b = 1 GbE 010b = 10 GbE 011b = 40 GbE All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RXCTL_CLK_SPEED	11:9	000b	RO	N/A	**RXCTL Clock Speed** Debug feature. Represents the current speed of the Rx clock for MAC 2. Speeds are represented as follows: Three bits per port (synchronized to the equivalent MAC clock): 000b = 100 Mb/s 001b = 1 GbE 010b = 10 GbE 011b = 40 GbE All other values are reserved.
UANA_CLK_SPEED	14:12	000b	RO	N/A	**UANA Clock Speed** Debug feature. Represents the current speed of the Rx clock for MAC 3. Speeds are represented as follows: Three bits per port (synchronized to the equivalent MAC clock): 000b = 100 Mb/s 001b = 1 GbE 010b = 10 GbE 011b = 40 GbE All other values are reserved.
RESERVED	17:15	000b	RSV	N/A	Reserved.
PE_CLK_SPEED	20:18	000b	RO	N/A	**PE Clock Speed**
RESERVED	31:21	0x0	RSV	N/A	Reserved.

13.2.2.1.41 Global Reset Status - GLGEN_RSTAT (0x000B8188; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DEVSTATE	1:0	00b	RO	N/A	**Device State** Device can be at one of the following states: 00b = Device active. 01b = Reset requested. 10b = Reset in progress. 11b = Reserved.
RESET_TYPE	3:2	00b	RO	N/A	**Reset Type** Reflects one of the following resets that are/were in progress: 00b = POR 01b = CORER 10b = GLOBR 11b = EMPR
CORERCNT	5:4	00b	RO	N/A	**Core Reset Count** Counts the number of initiated core resets since POR. The counter wraps around from 11b to 00b. ***Note:*** The counters count only the graceful resets asserted by the GLGEN_RTRIG register. It does not count resets initiated by the GLGEN_IMRTRIG register intended for internal use of the EMP.
GLOBRCNT	7:6	00b	RO	N/A	**Global Reset Count** Counts the number of initiated global resets since POR. The counter wraps around from 11b to 00b. ***Note:*** The counters count only the graceful resets asserted by the GLGEN_RTRIG register. It does not count resets initiated by the GLGEN_IMRTRIG register intended for internal use of the EMP.

Field	Bit(s)	Init.	Type	CFG Policy	Description
EMPRCNT	9:8	00b	RO	N/A	**EMP Reset Count** Counts the number of initiated EMP resets since POR. The counter wraps around from 11b to 00b. ***Note:*** The counters count only the graceful resets asserted by the GLGEN_RTRIG register. It does not count resets initiated by the GLGEN_IMRTRIG register intended for internal use of the EMP.
TIME_TO_RST	15:10	0x0	RO	N/A	**Time to Reset** The reset time is a down counter, loaded from GLGEN_RSTCTL.*GRSTDEL* following a GLOBR or CORER or EMPR. When *TIME_TO_RST* reaches a zero value the actual reset is initiated. This register is valid only in hardware-controlled resets (i.e., FW_RST_CONTROL_EN=0 or resets triggered by firmware AUX).
RTRIG_FLR	16	0b	RO	N/A	**Reset Trigger FLR** Reset request cause is due to FLR reset. Can only be asserted when switch mode is set.
RTRIG_ECC	17	0b	RO	N/A	**Reset Trigger ECC** Reset request cause is due to memories ECC-ERR. Set when GLGEN_RSTCTL_ECC_RST_ENA and memories uncorrectable error event accrued.
RTRIG_FW_AUX	18	0b	RO	N/A	**Reset Trigger Firmware AUX** Reset was triggered from firmware AUX register (for debug). This bit is hot ONLY - from AUX trigger until reset occurs.
RESERVED	31:19	0x0	RSV	N/A	Reserved.

13.2.2.1.42 Global Reset Trigger - GLGEN_RTRIG (0x000B8190; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CORER	0	0b	RW	UNDEFINED	**Core Reset** Setting this bit triggers a graceful core reset flow.
GLOBR	1	0b	RW	UNDEFINED	**Global Reset** Setting this bit triggers a graceful global reset flow.
RESERVED	31:2	0x0	RSV	N/A	Reserved.

13.2.2.1.43 Global Switch Mode Reset Control - GLGEN_ASSERT_HLP (0x000B81E4; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CORE_ON_RST	0	0b	RW	UNDEFINED	**Core on Reset** When set, indicates the device to assert HLP core reset on the next assertion of CORER. Asserts (hlpswitch_rst_b)
FULL_ON_RST	1	0b	RW	UNDEFINED	**Full on Reset** When set, indicates the device to assert HLP full reset on the next assertion of CORER. Asserts (hlpswitch_ports_rst_b)
RESERVED	31:2	0x0	RSV	N/A	Reserved.

13.2.2.1.44 Global Device Timer - GLVFGEN_TIMER (0x000B8214; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GTIME	31:0	0x0	RW	UNDEFINED	**GTIME** A free-running timer fed by a 1 μs clock.

13.2.2.1.45 Global Clock Status - GLGEN_CLKSTAT_SRC (0x000B826C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
U_CLK_SRC	1:0	00b	RO	N/A	**Upper Clock Source** Represents the current source of the respective core clock (core_clk) as follows: 00b = 367 MHz 01b = 416 MHz 10b = 446 MHz 11b = 396 MHz
L_CLK_SRC	3:2	00b	RO	N/A	**Lower Clock Source** Represents the current source of the respective core clock (core_clk) as follows: 00b = 367 MHz 01b = 416 MHz 10b = 446 MHz 11b = 396 MHz
PSM_CLK_SRC	5:4	00b	RO	N/A	**PSM Clock Source** Represents the current source of the respective core clock (core_clk) as follows: 00b = 367 MHz 01b = 416 MHz 10b = 446 MHz 11b = 396 MHz
RXCTL_CLK_SRC	7:6	00b	RO	N/A	**RXCTL Clock Source** Represents the current source of the respective core clock (core_clk) as follows: 00b = 367 MHz 01b = 416 MHz 10b = 446 MHz 11b = 396 MHz
UANA_CLK_SRC	11:8	0x0	RO	N/A	**UANA Clock Source** Represents the current source of the respective core clock (core_clk) as follows: 0x0 = 367 MHz 0x1 = 416 MHz 0x2 = 446 MHz 0x3 = 396 MHz 0x4 = 625 MHz
RESERVED	31:12	0x0	RSV	N/A	Reserved.

13.2.2.1.46 General Port Configuration3 - PRTGEN_CNF3 (0x000B8280; RO)

This register contains configuration per Ethernet port loaded from NVM.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PORT_STAGERING_EN	0	0b	RW	UNDEFINED	**Port Staggering Enable** The purpose of this bit is to allow firmware to stagger the enable of the PHY ports to prevent IR drop. On power-up reset, firmware staggers the enable of the ports that should be active during Sx mode. When moving to s0, it staggers all active ports. When going down to Sx, ports are down by hardware without staggering.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.1.47 LAN Port Number - PFGEN_PORTNUM (0x001D2400; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PORT NUM	2:0	000b	RW	UNDEFINED	**Port Number** Indicates the LAN port connected to this function.
RESERVED	31:3	0x0	RSV	N/A	Reserved.

13.2.2.1.48 GLGEN_ANA_FLAG_MAP - GLGEN_ANA_FLAG_MAP[n] (0x0020C000 + 0x4*n, n=0...63; RW)

Flags mapping register.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FLAG_EN	0	0b	RW	UNDEFINED	**Flag Enable** When set, Analyzer flag[N] should be exposed on flag *EXT_FLAG_ID* on the Analyzer outputs.
EXT_FLAG_ID	6:1	0x0	RW	UNDEFINED	**External Flag ID** CSR "i" maps the "i" external flag to *EXT_FLAG_ID* internal flag. Valid when *FLAG_EN* bit is set.
RESERVED	31:7	0x0	RSV	N/A	Reserved.

13.2.2.1.49 GLGEN_ANA_DEF_PTYPE - GLGEN_ANA_DEF_PTYPE (0x0020C100; RW)

Default PTYPE (when there is no match in PTYPE TCAM and node PTYPE is invalid).

Field	Bit(s)	Init.	Type	CFG Policy	Description
DEF_PTYPE	9:0	0x0	RW	UNDEFINED	**Default PTYPE** Analyzer default PTYPE value.
RESERVED	31:10	0x0	RSV	N/A	Reserved.

13.2.2.1.50 GLGEN_ANA_CFG_CTRL - GLGEN_ANA_CFG_CTRL (0x0020C104; RW)

Table's configuration control register.

Field	Bit(s)	Init.	Type	CFG Policy	Description
LINE_IDX	17:0	0x0	RW	UNDEFINED	**Line Index** Target memory line.
TABLE_ID	25:18	0x0	RW	UNDEFINED	**Table ID** Target table: 0x00 = TCAM key 0x01 = Xlate table (ALU) 0x02 = PG spill buffer action 0x03 = PG spill buffer key 0x04 = PG mem0 - holds all PG arches that sit at addr%8 0x05 = PG mem1 - holds all PG arches that sit at addr%8 + 1 0x06 = PG mem2 - holds all PG arches that sit at addr%8 + 2 0x07 = PG mem3 - holds all PG arches that sit at addr%8 + 3 0x08 = PG mem4 - holds all PG arches that sit at addr%8 + 4 0x09 = PG mem5 - holds all PG arches that sit at addr%8 + 5 0x0A = PG mem6 - holds all PG arches that sit at addr%8 + 6 0x0B = PG mem7 - holds all PG arches that sit at addr%8 + 7 0x0C = Protocol group memory 0x0D = TCAM action RAM 0x0E = Instruction memory 0x0F = Node cntx ID 0x10 = Marker group memory 0x11 = PTYPE TCAM key 0x12 = PTYPE TCAM action RAM 0x13 = No match PG spill buffer action 0x14 = No match PG spill buffer key 0x15 = No match PG mem0 - holds all PG arches that sit at addr%4 0x16 = No match PG mem1 - holds all PG arches that sit at addr%4 + 1 0x17 = No match PG mem2 - holds all PG arches that sit at addr%4 + 2 0x18 = No match PG mem3 - holds all PG arches that sit at addr%4 + 3 0x19 = Profiles table 0x1A = Profile entity mapping table
RESERVED	28:26	0x0	RW	N/A	Reserved.
OPERATION_ID	31:29	000b	RW	UNDEFINED	**Operation ID** 000b = WR 001b = RD 010b = Lookup for hit index (PG and no match PG) 011b = Lookup for free index (in PG and no match PG) All other values are reserved. To lookup PG, need to select the first memory of PG. To lookup NM PG, need to select the first memory of NM PG.

13.2.2.1.51 GLGEN_ANA_CFG_WRDATA - GLGEN_ANA_CFG_WRDATA (0x0020C108; RW)

Table's configuration write data register.

Field	Bit(s)	Init.	Type	CFG Policy	Description
WR_DATA	31:0	0x0	RW	UNDEFINED	**Write Data** 32-bit write data.

13.2.2.1.52 GLGEN_ANA_CFG_RDDATA - GLGEN_ANA_CFG_RDDATA[n] (0x0020C10C + 0x4*n, n=0...15; RO)

Table's configuration read data registers.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RD_DATA	31:0	0x0	RO	N/A	**Read Data** 32-bit read data.

13.2.2.1.53 GLGEN_ANA_CFG_LU_KEY - GLGEN_ANA_CFG_LU_KEY[n] (0x0020C14C + 0x4*n, n=0...2; RW)

Key to lookup CSR.

Field	Bit(s)	Init.	Type	CFG Policy	Description
LU_KEY	31:0	0x0	RW	UNDEFINED	**Lookup Key** KEY that is looked up when the lookup operations are selected.

13.2.2.1.54 GLGEN_ANA_CFG_HTBL_LU_RESULT - GLGEN_ANA_CFG_HTBL_LU_RESULT (0x0020C158; RO)

Lookup result from hash table.

Field	Bit(s)	Init.	Type	CFG Policy	Description
HIT	0	0b	RO	N/A	**Hit** For "look up for hit index" operation, this bit indicates if key is found in PG/NMPG tables. For "lookup for free index" operation, this bit indicates if there is a free entry in PG/NMPG tables.
PG_MEM_IDX	3:1	000b	RO	N/A	**PG Memory Index** For "look up for hit index" operation, this field indicates in which PG/NMPG memory the search key is found. For "lookup for free index" operation, this field indicates in which PG/NMPG memory the free entry is found.
ADDR	12:4	0x0	RO	N/A	**Address** For "look up for hit index" operation, this field indicates the memory address of the search key. For "lookup for free index" operation, this field indicates the memory address of the free entry.
RESERVED	31:13	0x0	RSV	N/A	Reserved.

13.2.2.1.55 GLGEN_ANA_CFG_SPLBUF_LU_RESULT - GLGEN_ANA_CFG_SPLBUF_LU_RESULT (0x0020C15C; RO)

Lookup result for spill buffer.

Field	Bit(s)	Init.	Type	CFG Policy	Description
HIT	0	0b	RO	N/A	**Hit** For "look up for hit index" operation, this bit indicates if key is found in PG/NMPG spill buffer. For "lookup for free index" operation, this bit indicates if there is a free entry in PG/NMPG spill buffer.
RESERVED	3:1	000b	RSV	N/A	Reserved.
ADDR	12:4	0x0	RO	N/A	**Address** For "look up for hit index" operation, this field indicates the spill buffer address of the search key. For "lookup for free index" operation, this field indicates the spill buffer address of the free entry.
RESERVED	31:13	0x0	RSV	N/A	Reserved.

13.2.2.1.56 GLGEN_ANA_P2P - GLGEN_ANA_P2P[n] (0x0020C160 + 0x4*n, n=0...15; RW)

Profile-to-profile mapping table.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	31:0	0x0	RSV	N/A	Reserved.

13.2.2.1.57 GLGEN_ANA_PG0_HASHKEY - GLGEN_ANA_PG0_HASHKEY[n] (0x0020C1A0 + 0x4*n, n=0...3; RW)

Main parse graph unit0 hash key registers.

Field	Bit(s)	Init.	Type	CFG Policy	Description
HASH_KEY	31:0	0x0	RW	UNDEFINED	**Hash Key** Bits [32*(N+1)-1:32*N] of the hash key of the first part of the primary parse graph.

13.2.2.1.58 GLGEN_ANA_NMPG0_HASHKEY - GLGEN_ANA_NMPG0_HASHKEY[n] (0x0020C1B0 + 0x4*n, n=0...3; RW)

"No match" parse graph unit0 hash key registers.

Field	Bit(s)	Init.	Type	CFG Policy	Description
HASH_KEY	31:0	0x0	RW	UNDEFINED	**Hash Key** Bits [32*(N+1)-1:32*N] of the hash key of the first part of the "no match" parse graph.

13.2.2.1.59 GLGEN_ANA_PG_KEYMASK - GLGEN_ANA_PG_KEYMASK[n] (0x0020C1C0 + 0x4*n, n=0...3; RW)

Main parse graph key mask registers.

Field	Bit(s)	Init.	Type	CFG Policy	Description
HASH_KEY	31:0	0x0	RW	UNDEFINED	**Hash Key** Bits [32*(N+1)-1:32*N] of the key mask vector of the primary parse graph. When mask=1, the corresponding bit in the key is cleared. Effective key = key & ~mask.

13.2.2.1.60 GLGEN_ANA_NMPG_KEYMASK - GLGEN_ANA_NMPG_KEYMASK[n] (0x0020C1D0 + 0x4*n, n=0...3; RW)

"No match" parse graph key mask registers.

Field	Bit(s)	Init.	Type	CFG Policy	Description
HASH_KEY	31:0	0x0	RW	UNDEFINED	**Hash Key** Bits [32*(N+1)-1:32*N] of the key mask vector of the "no match" parse graph. When mask=1, the corresponding bit in the key is cleared. Effective key = key & ~mask.

13.2.2.1.61 GLGEN_ANA_LAST_PROT_ID - GLGEN_ANA_LAST_PROT_ID[n] (0x0020C1E4 + 0x4*n, n=0...5; RW)

CSR for last override feature.

Field	Bit(s)	Init.	Type	CFG Policy	Description
EN	0	0b	RW	UNDEFINED	**Enable** Enable this protocol to be "last" protocol.
PROT ID	8:1	0x0	RW	UNDEFINED	**Protocol ID**
RESERVED	31:9	0x0	RSV	N/A	Reserved.

13.2.2.1.62 GLGEN_ANA_PROFIL - GLGEN_ANA_PROFIL_CTRL (0x0020C1FC; RW)

Number of profiles control register.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PROFILE_SELECT_MDID	4:0	0x0	RW	UNDEFINED	**Profile Select MDID** Used for extracting the "profile select entity" from input interfaces. This field describes the MDID from which the profile select entity is extracted.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PROFILE_SELECT_MDSTART	8:5	0x0	RW	UNDEFINED	**Profile Select MDID Start** The *MDstart* and *MDlen* fields define the "profile select entity" location in the MDID as follows: profile select entity = MDID[MDstart + MDlen -1 : MDID_start]
PROFILE_SELECT_MD_LEN	13:9	0x0	RW	UNDEFINED	**Profile Select MDID Length**
NUM_CTRL_DOMAIN	15:14	00b	RW	UNDEFINED	**Number of Control Domains** Defines the control domain key width in the PTYPE marker vector. 00b = 1 control domain (80 markers. No control domain bitmap.) 01b = 2 control domains (78 markers + 2b one-hot control domain bitmap). 10b = 4 control domains (76 markers + 4b one-hot control domain bitmap). 11b = 8 control domains (72 markers + 8b one-hot control domain bitmap).
DEF_PROF_ID	19:16	0x0	RW	UNDEFINED	**Default Profile ID** This profile is considered when the *SEL_DEF_PROF_ID* is set.
SEL_DEF_PROF_ID	20	0b	RW	UNDEFINED	**Select Default Profile ID** Select bit of the default Profile ID.
RESERVED	31:21	0x0	RSV	N/A	Reserved.

13.2.2.1.63 GLGEN_ANA_OUT_OF_PKT - GLGEN_ANA_OUT_OF_PKT (0x0020C200; RW)

"Header of out packet" error control register.

Field	Bit(s)	Init.	Type	CFG Policy	Description
NPC	7:0	0xBF	RW	UNDEFINED	**NPC** Error exception handling PC. Activated when NHO is advanced to a point outside packet after all headers are stored in buffer.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.1.64 GLGEN_ANA_NO_HIT_PG_NM_PG - GLGEN_ANA_NO_HIT_PG_NM_PG (0x0020C204; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
NPC	7:0	0xBF	RW	UNDEFINED	**NPC** Error exception handling PC. Activated when there is no hit either in PG or in no match PG.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.1.65 GLGEN_ANA_ALU_ACCSS_OUT_OF_PKT - GLGEN_ANA_ALU_ACCSS_OUT_OF_PKT (0x0020C208; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
NPC	7:0	0xBF	RW	UNDEFINED	**NPC** Error exception handling PC. Activated when ALU tries to access outside the last byte valid of the 256 bits of data available, or when PG does the same violation to build its key.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.1.66 GLGEN_ANA_INV_NODE_PTYPE - GLGEN_ANA_INV_NODE_PTYPE (0x0020C210; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
INV_NODE_PTYPE	10:0	0x0	RW	UNDEFINED	**Invalid Node PTYPE** Invalid node PTYPE number.
RESERVED	31:11	0x0	RSV	N/A	Reserved.

13.2.2.1.67 GLGEN_ANA_INV_PTYPE_MARKER - GLGEN_ANA_INV_PTYPE_MARKER (0x0020C218; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
INV_PTYPE_MARKER	6:0	0x7F	RW	UNDEFINED	**Invalid PTYPE Marker** Invalid PTYPE marker number.
RESERVED	31:7	0x0	RSV	N/A	Reserved.

13.2.2.1.68 GLGEN_ANA_ABORT_PTYPE - GLGEN_ANA_ABORT_PTYPE (0x0020C21C; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
ABORT	9:0	0xFF	RW	UNDEFINED	**Abort** Abort PTYPE number reflected on output interface when malicious packet is detected.
RESERVED	31:10	0x0	RSV	N/A	Reserved.

13.2.2.1.69 GLGEN_ANA_ERR_CTRL - GLGEN_ANA_ERR_CTRL (0x0020C220; RW)

Errors mask register.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ERR_MASK_EN	31:0	0xFFFFFFBF	RW	UNDEFINED	**Error Mask Enable** Per error type enable bit. When bit is cleared, error detection logic is not activated and error is not reported. Bit 0 = TCAM validity checks (for example: version field in ipv6 != 6). Bit 1 = HO is bigger pocket size. Bit 2 = There is no match in parse graph. Bit 3 = Illegal progress in parse graph (illegal leaf node). Bit 4 = HO is bigger than 504B (max header size). Bit 5 = Analyzer spent too many rounds on packet (number of cycles > fixed configured value). Bit 6 = Analyzer spent too many rounds on packet (number of cycles > packet cycles for performance). Bit 7 = ALU validity checks. Bit 8 = Packet has than 16 packet protocols Bit 9 = *total_ip* field not coherent to *pkt_len* field from interface. Bit 10 = ALU access outside the packet. Bit 11 = No hit in PG and no match PG.

13.2.2.1.70 GLGEN_ANA_TX_FLAG_MAP - GLGEN_ANA_TX_FLAG_MAP[n] (0x0020D000 + 0x4*n, n=0...63; RW)

Flags mapping register.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FLAG_EN	0	0b	RW	UNDEFINED	**Flag Enable** When set, Analyzer flag[N] should be exposed on flag *EXT_FLAG_ID* on the Analyzer outputs.
EXT_FLAG_ID	6:1	0x0	RW	UNDEFINED	**External Flag ID** CSR "i" maps the "i" external flag to *EXT_FLAG_ID* internal flag. Valid when *FLAG_EN* bit is set.
RESERVED	31:7	0x0	RSV	N/A	Reserved.

13.2.2.1.71 GLGEN_ANA_TX_DEF_PTYPE - GLGEN_ANA_TX_DEF_PTYPE (0x0020D100; RW)

Default PTYPE (when there is no match in PTYPE TCAM and node PTYPE is invalid).

Field	Bit(s)	Init.	Type	CFG Policy	Description
DEF_PTYPE	9:0	0x0	RW	UNDEFINED	**Default PTYPE** Analyzer default PTYPE value.
RESERVED	31:10	0x0	RSV	N/A	Reserved.

13.2.2.1.72 GLGEN_ANA_TX_CFG_CTRL - GLGEN_ANA_TX_CFG_CTRL (0x0020D104; RW)

Table's configuration control register.

Field	Bit(s)	Init.	Type	CFG Policy	Description
LINE_IDX	17:0	0c0	RW	UNDEFINED	**Line Index** Target memory line.
TABLE_ID	25:18	0x0	RW	UNDEFINED	**Table ID** Target table: 0x00 = TCAM key 0x01 = Xlate table (ALU) 0x02 = PG spill buffer action 0x03 = PG spill buffer key 0x04 = PG mem0 - Holds all PG arches that sit at addr%8 0x05 = PG mem1 - Holds all PG arches that sit at addr%8 + 1 0x06 = PG mem2 - Holds all PG arches that sit at addr%8 + 2 0x07 = PG mem3 - Holds all PG arches that sit at addr%8 + 3 0x08 = PG mem4 - Holds all PG arches that sit at addr%8 + 4 0x09 = PG mem5 - Holds all PG arches that sit at addr%8 + 5 0x0A = PG mem6 - Holds all PG arches that sit at addr%8 + 6 0x0B = PG mem7 - Holds all PG arches that sit at addr%8 + 7 0x0C = Protocol group memory 0x0D = TCAM action RAM 0x0E = Instruction memory 0x0F = Node cntx ID 0x10 = marker group memory 0x11 = PTYPE TCAM key 0x12 = PTYPE TCAM action RAM 0x13 = no match PG spill buffer action 0x14 = no match PG spill buffer key 0x15 = no match PG mem0- Holds all PG arches that sit at addr%4 0x16 = no match PG mem1 - Holds all PG arches that sit at addr%4 + 1 0x17 = no match PG mem2 - Holds all PG arches that sit at addr%4 + 2 0x18 = no match PG mem3 - Holds all PG arches that sit at addr%4 + 3 0x19 = profiles table 0x1A = profile entity mapping table
RESERVED	28:26	000b	RSV	N/A	Reserved.
OPERATION _ID	31:29	000b	RW	UNDEFINED	**Operation ID** 000b = WR 001b = RD 010b = Lookup for hit index (PG and no match PG) 011b = Lookup for free index (in PG and no match PG) All other values are reserved. To lookup PG, need to select the first memory of PG. To lookup NM PG, need to select the first memory of NM PG.

13.2.2.1.73 GLGEN_ANA_TX_CFG_WRDATA - GLGEN_ANA_TX_CFG_WRDATA (0x0020D108; RW)

Table's configuration write data register.

Field	Bit(s)	Init.	Type	CFG Policy	Description
WR_DATA	31:0	0x0	RW	UNDEFINED	**Write Data** 32-bit write data.

13.2.2.1.74 GLGEN_ANA_TX_CFG_RDDATA - GLGEN_ANA_TX_CFG_RDDATA[n] (0x0020D10C + 0x4*n, n=0...15; RO)

Table's configuration read data registers.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RD_DATA	31:0	0x0	RO	N/A	**Read Data** 32-bit read data.

13.2.2.1.75 GLGEN_ANA_TX_CFG_LU_KEY - GLGEN_ANA_TX_CFG_LU_KEY[n] (0x0020D14C + 0x4*n, n=0...2; RW)

Key to lookup CSR.

Field	Bit(s)	Init.	Type	CFG Policy	Description
LU_KEY	31:0	0x0	RW	UNDEFINED	**Lookup Key** KEY that is looked up when the lookup operations are selected.

13.2.2.1.76 GLGEN_ANA_TX_CFG_HTBL_LU_RESULT - GLGEN_ANA_TX_CFG_HTBL_LU_RESULT (0x0020D158; RO)

Lookup result from hash table.

Field	Bit(s)	Init.	Type	CFG Policy	Description
HIT	0	0b	RO	N/A	**Hit** For "look up for hit index" operation, this bit indicates if key is found in PG/NMPG tables. For "lookup for free index" operation, this bit indicates if there is a free entry in PG/NMPG tables.
PG_MEM_IDX	3:1	000b	RO	N/A	**PG Memory Index** For "look up for hit index" operation, this field indicates in which PG/NMPG memory the search key is found. For "lookup for free index" operation, this field indicates in which PG/NMPG memory the free entry is found.
ADDR	12:4	0x0	RO	N/A	**Address** For "look up for hit index" operation, this field indicates the memory address of the search key. For "lookup for free index" operation, this field indicates the memory address of the free entry.
RESERVED	31:13	0x0	RSV	N/A	Reserved.

13.2.2.1.77 GLGEN_ANA_TX_CFG_SPLBUF_LU_RESULT - GLGEN_ANA_TX_CFG_SPLBUF_LU_RESULT (0x0020D15C; RO)

Lookup result for spill buffer.

Field	Bit(s)	Init.	Type	CFG Policy	Description
HIT	0	0b	RO	N/A	**Hit** For "look up for hit index" operation, this bit indicates if key is found in PG/NMPG spill buffer. For "lookup for free index" operation, this bit indicates if there is a free entry in PG/NMPG spill buffer.
RESERVED	3:1	000b	RSV	N/A	Reserved.
ADDR	12:4	0x0	RO	N/A	**Address** For "look up for hit index" operation, this field indicates the spill buffer address of the search key. For "lookup for free index" operation, this field indicates the spill buffer address of the free entry.
RESERVED	31:13	0x0	RSV	N/A	Reserved.

13.2.2.1.78 GLGEN_ANA_TX_P2P - GLGEN_ANA_TX_P2P[n] (0x0020D160 + 0x4*n, n=0...15; RW)

Profile-to-profile mapping table.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	31:0	0x0	RSV	N/A	Reserved.

13.2.2.1.79 GLGEN_ANA_TX_PG0_HASHKEY - GLGEN_ANA_TX_PG0_HASHKEY[n] (0x0020D1A0 + 0x4*n, n=0...3; RW)

Main parse graph unit0 hash key registers.

Field	Bit(s)	Init.	Type	CFG Policy	Description
HASH_KEY	31:0	0x0	RW	UNDEFINED	**Hash Key** Bits [32*(N+1)-1:32*N] of the hash key of the first part of the primary parse graph.

13.2.2.1.80 GLGEN_ANA_TX_NMPG0_HASHKEY - GLGEN_ANA_TX_NMPG0_HASHKEY[n] (0x0020D1B0 + 0x4*n, n=0...3; RW)

"No match" parse graph unit0 hash key registers.

Field	Bit(s)	Init.	Type	CFG Policy	Description
HASH_KEY	31:0	0x0	RW	UNDEFINED	**Hash Key** Bits [32*(N+1)-1:32*N] of the hash key of the first part of the "no match" parse graph.

13.2.2.1.81 GLGEN_ANA_TX_PG_KEYMASK - GLGEN_ANA_TX_PG_KEYMASK[n] (0x0020D1C0 + 0x4*n, n=0...3; RW)

Main parse graph key mask registers.

Field	Bit(s)	Init.	Type	CFG Policy	Description
HASH_KEY	31:0	0x0	RW	UNDEFINED	**Hash Key** Bits [32*(N+1)-1:32*N] of the key mask vector of the primary parse graph. When mask=1, the corresponding bit in the key is cleared. Effective key = key & ~mask.

13.2.2.1.82 GLGEN_ANA_TX_NMPG_KEYMASK - GLGEN_ANA_TX_NMPG_KEYMASK[n] (0x0020D1D0 + 0x4*n, n=0...3; RW)

"No match" parse graph key mask registers.

Field	Bit(s)	Init.	Type	CFG Policy	Description
HASH_KEY	31:0	0x0	RW	UNDEFINED	**Hash Key** Bits [32*(N+1)-1:32*N] of the key mask vector of the "no match" parse graph. When mask=1, the corresponding bit in the key is cleared. Effective key = key & ~mask.

13.2.2.1.83 GLGEN_ANA_TX_PROFIL_CTRL - GLGEN_ANA_TX_PROFIL_CTRL (0x0020D1FC; RW)

Number of profiles control register.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PROFILE_SELECT_MDID	4:0	0x0	RW	UNDEFINED	**Profile Select MDID** Used for extracting the "profile select entity" from input interfaces. This field describes the MDID from which the profile select entity is extracted.
PROFILE_SELECT_MDSTART	8:5	0x0	RW	UNDEFINED	**Profile Select MDID Start** The *MDstart* and *MDlen* fields define the "profile select entity" location in the MDID as follows: profile select entity = MDID[MDstart + MDlen -1 : MDID_start]
PROFILE_SELECT_MD_LEN	13:9	0x0	RW	UNDEFINED	**Profile Select MDID Length**
NUM_CTRL_DOMAIN	15:14	00b	RW	UNDEFINED	**Number of Control Domains** Defines the control domain key width in the PTYPE marker vector. 00b = 1 control domain (80 markers. No control domain bitmap.) 01b = 2 control domains (78 markers + 2b one-hot control domain bitmap). 10b = 4 control domains (76 markers + 4b one-hot control domain bitmap). 11b = 8 control domains (72 markers + 8b one-hot control domain bitmap).

Field	Bit(s)	Init.	Type	CFG Policy	Description
DEF_PROF_ID	19:16	0x0	RW	UNDEFINED	**Default Profile ID** This profile is considered when the *SEL_DEF_PROF_ID* is set.
SEL_DEF_PROF_ID	20	0b	RW	UNDEFINED	**Select Default Profile ID** Select bit of the default Profile ID.
RESERVED	31:21	0x0	RSV	N/A	Reserved.

13.2.2.1.84 GLGEN_ANA_TX_NO_HIT_PG_NM_PG - GLGEN_ANA_TX_NO_HIT_PG_NM_PG (0x0020D204; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
NPC	7:0	0xBF	RW	UNDEFINED	**NPC** Error exception handling PC. Activated when there is no hit either in PG or in no match PG.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.1.85 GLGEN_ANA_TX_ALU_ACCSS_OUT_OF_PKT - GLGEN_ANA_TX_ALU_ACCSS_OUT_OF_PKT (0x0020D208; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
NPC	7:0	0xBF	RW	UNDEFINED	**NPC** Error exception handling PC. Activated when ALU tries to access outside the last byte valid of the 256b of data available or when PG does the same violation to build its key.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.1.86 GLGEN_ANA_TX_INV_NODE_PTYPE - GLGEN_ANA_TX_INV_NODE_PTYPE (0x0020D210; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
INV_NODE_PTYPE	10:0	0x0	RW	UNDEFINED	**Invalid Node PTYPE** Invalid node PTYPE number.
RESERVED	31:11	0x0	RSV	N/A	Reserved.

13.2.2.1.87 GLGEN_ANA_TX_INV_PROT_ID - GLGEN_ANA_TX_INV_PROT_ID (0x0020D214; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
INV_PROT_ID	7:0	0xFF	RW	UNDEFINED	**Invalid Protocol ID** Invalid protocol ID.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.1.88 GLGEN_ANA_TX_INV_PTYPE_MARKER - GLGEN_ANA_TX_INV_PTYPE_MARKER (0x0020D218; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
INV_PTYPE_MARKER	6:0	0x7F	RW	UNDEFINED	**Invalid PTYPE Marker** Invalid PTYPE marker number.
RESERVED	31:7	0x0	RSV	N/A	Reserved.

13.2.2.1.89 GLGEN_ANA_TX_ABORT_PTYPE - GLGEN_ANA_TX_ABORT_PTYPE (0x0020D21C; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
ABORT	9:0	0xFF	RW	UNDEFINED	**Abort** Abort PTYPE number reflected on output interface when malicious packet is detected.
RESERVED	31:10	0x0	RSV	N/A	Reserved.

13.2.2.1.90 GLGEN_ANA_TX_ERR_CTRL - GLGEN_ANA_TX_ERR_CTRL (0x0020D220; RW)

Errors mask register.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ERR_MASK_EN	31:0	0xFFFFFFBF	RW	UNDEFINED	**Error Mask Enable** Per error type enable bit. When bit is cleared, error detection logic is not activated and error is not reported. Bit 0 = TCAM validity checks (for example: version field in ipv6 != 6). Bit 1 = HO is bigger pocket size. Bit 2 = There is no match in parse graph. Bit 3 = Illegal progress in parse graph (illegal leaf node). Bit 4 = HO is bigger than 504B (max header size). Bit 5 = Analyzer spent too many rounds on packet (number of cycles > fixed configured value). Bit 6 = Analyzer spent too many rounds on packet (number of cycles > packet cycles for performance). Bit 7 = ALU validity checks. Bit 8 = Packet has than 16 packet protocols Bit 9 = *total_ip* field not coherent to *pkt_len* field from interface. Bit 10 = ALU access outside the packet. Bit 11 = No hit in PG and no match PG.

13.2.2.1.91 GLGEN_ANA_TX_DFD_PACE_OUT - GLGEN_ANA_TX_DFD_PACE_OUT (0x0020D4CC; RW)

DFD pacing.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PUSH	0	0b	SC	UNDEFINED	**Push** Pace the output of the analyzer.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.2 PF - Internal Fuses Registers

13.2.2.2.1 SKU Fuses - GL_UFUSE_SOC (0x000A400C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PORT_MODE	1:0	00b	RO	N/A	**Port Mode** Bit 1:0 in fuse map in Pull Fuse message. 00b = Octal Port 01b = Quad Port 10b = Dual Port 11b = Single Port
BANDWIDTH	3:2	00b	RO	UNDEFINED	**Bandwidth** Bit 3:2 in fuse map in Pull Fuse message. Maximum network speed: 00b = 200 Gb/s 01b = 100 Gb/s 10b = 50 Gb/s 11b = 25 Gb/s
PE_DISABLE	4	0b	RO	UNDEFINED	**PE Disable** Bit 4 in fuse map in Pull Fuse message. Controls the enablement of the PE engine: 0b = PE enabled. 1b = PE disabled.
SWITCH_MODE	5	0b	RO	UNDEFINED	**Switch Mode** Bit 5 in fuse map in Pull Fuse message. Switch mode enable: 0b = Switch mode enabled. 1b = Switch mode disabled.
CSR_PROTECTION_ENABLE	6	0b	RO	UNDEFINED	**CSR Protection Enable** Bit 6 in fuse map in Pull Fuse message. 0b = CSR Protection disabled. 1b = CSR Protection enabled.
RESERVED	8:7	0x0	RSV	N/A	Reserved.
BLOCK_BME_TO_FW	9	0b	RO	UNDEFINED	**Block BME Access to Firmware** Bit 9 in fuse map in Pull Fuse message. 0b = BME is writable by firmware. 1b = BME is not writable by firmware.
SOC_TYPE	10	0b	RO	UNDEFINED	**SoC Type** Bit 10 in fuse map in Pull Fuse message. 0b = ICX-D 1b = SNR
BTS_MODE	11	0b	RO	UNDEFINED	**BTS Mode** Bit 11 in fuse map in Pull Fuse message. 0b = BTS mode. 1b = Non-BTS mode.
RESERVED	31:12	0x0000F	RSV	N/A	Reserved.

13.2.2.3 PF - PCIe Registers

This category contains registers for PCIe configuration and control.

13.2.2.3.1 PFPCIe Subsystem ID - PFPCI_SUBSYSID (0x0009D880; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PF_SUBSYS_ID	15:0	0x0	RW	UNDEFINED	**PF Subsystem ID** Subsystem ID for this PF.
VF_SUBSYS_ID	31:16	0x0	RW	UNDEFINED	**VF Subsystem ID** Subsystem ID for VFs of this PF.

13.2.2.3.2 PCIe Functions Configuration - PFPCI_FUNC (0x0009D980; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FUNC_DIS	0	1b	RW	UNDEFINED	**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.
ALLOW_FUNC_DIS	1	0b	RW	UNDEFINED	**Allow Function Disable** 0b = Asserting PCI_DIS_N has no effect on this PCI function. 1b = Asserting PCI_DIS_N disables this PCI function.
DIS_FUNC_ON_PORT_DIS	2	0b	RW	UNDEFINED	**Disable Function on Port Disable** 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.
RESERVED	31:3	0x0	RSV	N/A	Reserved.

13.2.2.3.3 PCIe Function Status 1 - PFPCI_STATUS1 (0x0009DA00; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FUNC_VALID	0	0b	RO	N/A	**Function 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	N/A	Reserved.

13.2.2.3.4 PCIe PM - PFPCI_PM (0x0009DA80; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PME_EN	0	0b	RW	UNDEFINED	**PME Enable** This read/write bit is used by the software device driver to generate a PME event without writing to the Power Management Control/Status Register (PMCSR) in the PCIe configuration space. ***Note:*** The internal PME Enablement is a logic OR function of the following: *PME_EN* flag in the PMCSR, *PME_EN* flag in the PFPCI_PM CSR, and *APME* flag in the PFPM_APM CSR. The bit is reset on STRST (Sticky Reset): bit is reset only on power-on reset (LAN_PWR_GOOD). When *AUX_PWR* = 0b, this bit is also reset when de-asserting PE_RST_N.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.3.5 PCIe Storage Class - PFPCI_CLASS (0x0009DB00; RO)

Contains per-PF configuration loaded from NVM.

Field	Bit(s)	Init.	Type	CFG Policy	Description
STORAGE_CLASS	0	0b	RW	UNDEFINED	**Storage Class** 0b = The class code of this function is set to 0x020000 (LAN). 1b = The class code of this function is set to 0x010000 (SCSI).
RESERVED	1	0b	RSV	N/A	Reserved.
PF_IS_LAN	2	1b	RW	UNDEFINED	**PF is LAN** 0b = SAN function. 1b = LAN function.
RESERVED	31:3	0x0	RSV	N/A	Reserved.

13.2.2.3.6 PCIe PF Device ID - PFPCI_DEVID (0x0009DE00; RO)

Contains the per-PF Device ID.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PF_DEV_ID	15:0	0x374C	RW	UNDEFINED	**PF Device ID** Contains the device ID for this PF.
VF_DEV_ID	31:16	0x374D	RW	UNDEFINED	**VF Device ID** Contains the device ID for the VFs of this PF.

13.2.2.3.7 Clock Gating Events - GL_CLKGATE_EVENTS (0x0009DE70; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRIMARY_CLKGATE_EVENTS	15:0	0x0	RO	N/A	**Primary Clock Gating Events** P-IOSF Clock gating aliveness indication (increments every clock gating event - wraps around).
SIDEBAND_CLKGATE_EVENTS	31:16	0x0	RO	N/A	**Sideband Clock Gating Events** SB-IOSF Clock gating aliveness indication (increments every clock gating event - wraps around).

13.2.2.3.8 PCI BAR Control - GLPCI_LBARCTRL (0x0009DE74; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PREFBAR	0	1b	RW	UNDEFINED	**Prefetchable BAR** Prefetchable bit indication in the memory BAR and MSI-X BAR (should be set when 64-bit BARs are used). 0b = BARs are marked as non prefetchable. 1b = BARs are marked as prefetchable.
RESERVED	1	0b	RSV	N/A	Reserved.
PAGES_SPACE_EN_PF	2	0b	RW	UNDEFINED	**Pages Space Enable PF** When set, the PF switch mode memory space is accessible through the memory BAR
FLASH_EXPOSE	3	1b	RW	UNDEFINED	**Flash Exposed** When set, the Flash memory is accessible through the memory BAR.
PE_DB_SIZE	5:4	00b	RW	UNDEFINED	**PE Doorbell Size** Determines the size of the memory space allocated to the protocol engine doorbells in the PF BARs: 00b = Memory space is not allocated for PE doorbells. 01b = A 64 KB area is allocated. 10b = A (8 MB + 64 KB) area is allocated. 11b = Reserved.
RESERVED	8:6	111b	RSV	N/A	Reserved.
PAGES_SPACE_EN_VF	9	0b	RW	UNDEFINED	**Pages Space Enable PF** When set, the VF switch mode memory space is accessible through the memory BAR
RESERVED	10	0b	RSV	N/A	Reserved.
EXROM_SIZE	13:11	011b	RW	UNDEFINED	**Expansion ROM Size** This field indicates the size of the Expansion ROM BAR as = 64 KB x (2 ** EXROM_SIZE). 000b = 64 KB size. ... 111b = 8 MB size. Default value is 512 KB.
VF_PE_DB_SIZE	15:14	00b	RW	UNDEFINED	**VF PE Doorbell Size** Determines the size of the memory space allocated to the protocol engine doorbells in the VF BARs: 00b = Memory Space is not allocated for PE doorbells. 01b = An 8 KB area is allocated right after the legacy space (0x10000 ->0x11FFF). 10b = A 64 KB area is allocated (0x10000 ->0x20000). 11b = Reserved.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.3.9 PCIe Power Data Register - GLPCI_PWRDATA (0x0009DE7C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
D0_POWER	7:0	0x0	RW	UNDEFINED	**D0 Power** 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).

Field	Bit(s)	Init.	Type	CFG Policy	Description
COMM_POWER	15:8	0x0	RW	UNDEFINED	**Common Power** 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	UNDEFINED	**D3 Power** 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	00b	RW	UNDEFINED	**Data Scale** The value in this field reflects the *Data_Scale* field in the PCI PMCSR register.
RESERVED	31:26	0x0	RSV	N/A	Reserved.

13.2.2.3.10 PCIe Serial Number MAC Address Low - GLPCI_SERL (0x0009DE80; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SER_NUM_L	31:0	0x0	RW	UNDEFINED	**Serial Number Low** The low DWord of the Ethernet MAC Address used to generate the PCIe serial number. The register contents is loaded from NVM. The location in NVM is pointed by the fourth item in the Auto-Generated Pointers Module. It is a per-device manufacturing value that represents the whole device. It can be set identical to the concatenated [PFPM_SAL1|PFPM_SAL0] words of the EMP Settings Module.

13.2.2.3.11 PCIe Serial Number MAC Address High - GLPCI_SERH (0x0009DE84; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SER_NUM_H	15:0	0x0	RW	UNDEFINED	**Serial Number High** The high word of the Ethernet MAC Address used to generate the PCIe serial number. The register contents is loaded from NVM. The location in NVM is pointed by the fifth item in the Auto-Generated Pointers Module. It is a per device manufacturing value which represents the whole device. It can be set identical to the concatenated [PFPM_SAH1|PFPM_SAH0] words of the EMP Settings Module.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.3.12 PCIe Capabilities Control - GLPCI_CAPCTRL (0x0009DE88; RW)

Determines PCIe capabilities supported by the device and that software is allowed to enable or disable.

Field	Bit(s)	Init.	Type	CFG Policy	Description
VPD_EN	0	0b	RW	UNDEFINED	**VPD Enable** 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	N/A	Reserved.

13.2.2.3.13 PCIe Capabilities Support - GLPCI_CAPSUP (0x0009DE8C; RO)

Determines PCIe capabilities supported by the device.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PCIE_VER	0	1b	RW	UNDEFINED	**PCIe Version** Determines the PCIe capability version. 0b = Capability version 0x1. 1b = Capability version 0x2.
RESERVED	1	0b	RSV	N/A	Reserved.
LTR_EN	2	0b	RW	UNDEFINED	**LTR Enable** 1b = Indicates support for PCIe Latency Tolerance Reporting (LTR) capability. This bit must be set to 0b (LTR is not supported by this product).
TPH_EN	3	1b	RW	UNDEFINED	**TPH Enable** 1b = Indicates support for the PCIe TPH requester capability.
ARI_EN	4	1b	RW	UNDEFINED	**ARI Enable** 1b = Indicates support for PCIe ARI capability.
IOV_EN	5	1b	RW	UNDEFINED	**IOV Enable** 1b = Indicates support for PCIe SR-IOV capability.
ACS_EN	6	1b	RW	UNDEFINED	**ACS Enable** 1b = Indicates support for PCIe ACS capability.
SEC_EN	7	1b	RW	UNDEFINED	**Secondary Enable** 1b = Indicates support for the secondary PCIe extended capability.
PASID_EN	8	0b	RW	UNDEFINED	**PASID Enable** 1b = Indicates support for PCIe PASID capability.
DLFE_EN	9	0b	RW	UNDEFINED	**Data Link Feature Enable** 1b = Indicates support for PCIe Data Link Feature capability.
GEN4_EXT_EN	10	0b	RW	UNDEFINED	**Gen4 Extended Enable** 1b = Indicates support for PCIe physical layer 13.0 GT/s extended capability.
GEN4_MARG_EN	11	0b	RW	UNDEFINED	**Gen4 Margining Enable** 1b = Indicates support for PCIe lane margining at receiver capability.
RESERVED	15:12	0x0	RSV	N/A	Reserved.
ECRC_GEN_EN	16	1b	RW	UNDEFINED	**ECRC Generation Enable** Loaded into the ECRC. Generation capable bit of the PCIe configuration registers.
ECRC_CHK_EN	17	1b	RW	UNDEFINED	**ECRC Check Enable** Loaded into the ECRC Check capable bit of the PCIe configuration registers.
IDO_EN	18	1b	RW	UNDEFINED	**IDO Enable** Enables ID-based ordering (IDO).
MSI_MASK	19	1b	RW	UNDEFINED	**MSI Mask** 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	UNDEFINED	**CSR Configuration Enable** Enables access to CSRs via the PCI configuration space. See the section on Configuration Access to Internal Registers and Memories.

Field	Bit(s)	Init.	Type	CFG Policy	Description
WAKUP_EN	21	0b	RW	UNDEFINED	**Wake-Up Enable** 1b = Wake-up in D3 is exposed in *PME_Support* field in Power Management Capabilities register
RESERVED	29:22	0x0	RSV	N/A	Reserved.
LOAD_SUBSYS_ID	30	0b	RW	UNDEFINED	**Load Subsystem IDs** 1b = Indicates that the device loads its PCIe sub-system ID and sub-system vendor ID from the NVM.
LOAD_DEV_ID	31	0b	RW	UNDEFINED	**Load Device ID** 1b = Indicates that the device loads its PCI device IDs from the NVM.

13.2.2.3.14 PCIe Link Capabilities - GLPCI_LINKCAP (0x0009DE90; RO)

Determines PCIe link capabilities supported by the device.

Field	Bit(s)	Init.	Type	CFG Policy	Description
LINK_SPEEDS_VECTOR	5:0	0x0	RW	UNDEFINED	**Supported Link Speeds Vector** Loaded to the Link Capabilities 2 register in the PCIe capability. Bit 0 = 5.0 GT/s Bit 1 = 8.0 GT/s Bits 5:2 = Reserved ***Note:*** 2.5 GT/s is always supported. Must be zero.
RESERVED	8:6	100b	RSV	N/A	Reserved.
MAX_LINK_WIDTH	12:9	0x7	RW	UNDEFINED	**Max Link Width** Loaded to the PCIe Link Capabilities register. 0x1 = Limit max link width to x1. 0x3 = Limit max link width to x4. 0x4 = Limit max link width to x8. 0x7 = Do not limit max link width. Negotiate to the max width supported by the link. All other values are reserved. Must be 0001b
RESERVED	31:13	0x0	RSV	N/A	Reserved.

13.2.2.3.15 PCIe PM Support - GLPCI_PMSUP (0x0009DE94; RO)

This register contains parameters that define PCIe power management support.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	7:0	0x97	RSV	N/A	Reserved.
L0S_ACC_LAT	10:8	011b	RW	UNDEFINED	**L0s Acceptable Latency** Loaded to the *Endpoint L0s Acceptable Latency* field in the PCIe Device Capabilities register.
L1_ACC_LAT	13:11	110b	RW	UNDEFINED	**L1s Acceptable Latency** Loaded to the *Endpoint L1 Acceptable Latency* field in the PCIe Device Capabilities register.
RESERVED	14	1b	RSV	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
OBFF_SUP	16:15	00b	RW	UNDEFINED	**OBFF Supported** 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	N/A	Reserved.

13.2.2.3.16 PCIe Revision ID - GLPCI_REVID (0x0009DE98; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
NVM_REVID	7:0	0x0	RW	UNDEFINED	**NVM Revision ID** Value of the Rev ID loaded from the NVM. The actual value reflected in the config space is the XOR of this field with the GLPCI_DREVID.*DEFAULT_REVID* value.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.3.17 PCIe VF Capabilities Support - GLPCI_VFSUP (0x0009DE9C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VF_PREFETCH	0	1b	RW	UNDEFINED	**VF Prefetchable** 0b = IOV memory BAR and MSI-X BAR are declared as non-prefetchable. 1b = IOV memory BAR and MSI-X BAR are declared as prefetchable.
RESERVED	31:1	0x1	RSV	N/A	Reserved.

13.2.2.3.18 PCIe Global Config - GLPCI_CNF (0x0009DEA0; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	1:0	00b	RSV	N/A	Reserved.
WAKE_PIN_EN	2	0b	RW	UNDEFINED	**WAKE Pin Enable** When set to 1b, enables the use of the WAKE pin for a PME event in all power states.
MSIX_ECC_BLOCK_DISABLE	3	0b	RW	UNDEFINED	**MSI-X ECC Block Disable** When set, MSI-X vector tables is not blocked after an ECC error event. This bit can also be used to release the MSI-X table as part of an ECC recovery flow by setting it and then clearing it again.
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.3.19 PCIe Vendor ID - GLPCI_VENDORID (0x0009DEC8; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VENDORID	15:0	0x8086	RW	UNDEFINED	**Vendor ID** 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	N/A	Reserved.

13.2.2.3.20 PCIe Subsystem ID - GLPCI_SUBVENID (0x0009DEE8; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SUB_VEN_ID	15:0	0x8086	RW	UNDEFINED	**Subsystem Vendor ID** Loaded to the PCI configuration Subsystem Vendor ID register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.3.21 PCIe PF Configuration - PFPCI_CNF (0x0009DF00; RO)

Contains per-PF configuration loaded from NVM.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	1:0	00b	RSV	N/A	Reserved.
MSI_EN	2	1b	RW	UNDEFINED	**MSI Enable** Enables the MSI capability structure for this PCI function. 0b = MSI is disabled. 1b = MSI is enabled.
EXROM_DIS	3	0b	RW	UNDEFINED	**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	UNDEFINED	**I/O BAR Support** 0b = I/O BAR is not supported. 1b = I/O BAR is supported.
INT_PIN	6:5	00b	RW	UNDEFINED	**Interrupt Pin** Controls the value advertised in the *Interrupt Pin* field of the PCI configuration header for this function. 00b = INTA# 01b = INTB# 10b = INTC# 11b = INTD# The value advertised in the PCI configuration header is the value loaded from NVM + 1.
RESERVED	31:7	0x0	RSV	N/A	Reserved.

13.2.2.3.22 Posted Queue IOSF FIFO Status - PQ_FIFO_STATUS (0x0009DF40; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PQ_FIFO_COUNT	30:0	0x0	RO	N/A	**Posted Queue FIFO Count** Counter of PFB POSTED FIFO. Increases by 1 when the FIFO round is completed.
PQ_FIFO_EMPTY	31	1b	RO	UNDEFINED	**Posted Queue FIFO Empty** PFB POSTED FIFO is empty.

13.2.2.3.23 Push PE IF Status - GLPCI_PUSH_PE_IF_TO_STATUS (0x0009DF44; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GLPCI_PUSH_PE_IF_TO_STATUS	0	0b	RW1C	UNDEFINED	**Push PE Interface Timeout Status** Once time-out is reached, the PCIe Push logic silently drains the push buffer in PCIe. Exiting this mechanism is done by CORER or writing 1b.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.3.24 PCIe VF Flush Done - PFPCI_VF_FLUSH_DONE[VF] (0x0009E000 + 0x4*VF, VF=0...255; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FLUSH_DONE	0	0b	RO	N/A	**Flush Done** VF transaction pending bit. Reset by function level reset, and set when the pipe is clean.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.3.25 PCIe PF Flush Done - PFPCI_PF_FLUSH_DONE (0x0009E400; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FLUSH_DONE	0	0b	RO	N/A	**Flush Done** PF transaction pending bit. Reset by function level reset, and set when the pipe is clean.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.3.26 PCIe VM Flush Done - PFPCI_VM_FLUSH_DONE (0x0009E480; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FLUSH_DONE	0	0b	RO	N/A	**Flush Done** VM transaction pending bit. Reset by function level reset, and set when the pipe is clean.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.3.27 PCIe Configuration Indirect Access Data - PF_PCI_CIAD (0x0009E500; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DATA	31:0	0x0	RW	UNDEFINED	**Data** Used to access the configuration registers of the VF. It operates together with the PF_PCI_CIAA register as follows: • Reading this register returns the content of the register at offset = *ADDRESS* in the configuration space of VF index = *VF_NUM* (the ADDRESS and VF_NUM parameters are defined by the PF_PCI_CIAA register). • Writing to this register is gated by the *CONFIG_ACCESS_ENABLE* flag in the GL_PCI_DBGCTL register. If enabled, the value written to this register is programmed to the register at offset = *ADDRESS* in the configuration space of VF index = *VF_NUM*.

13.2.2.3.28 PF Configuration Indirect Access Address - PF_PCI_CIAA (0x0009E580; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
ADDRESS	11:0	0x0	RW	UNDEFINED	**Address** The configuration space address to access.
VF_NUM	19:12	0x0	RW	UNDEFINED	**VF Number** Defines the VF number to access. The VF number is the absolute VF number in the device. ***Note:*** The PF can access any VF (its own VFs as well as other VFs).
RESERVED	31:20	0x0	RSV	N/A	Reserved. Ignore on read. Write 0b.

13.2.2.3.29 PCIe VM Pending Index - PFPCI_VMINDEX (0x0009E600; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VMINDEX	9:0	0x0	RW	UNDEFINED	**VM Index** Software sets the VMINDEX that its transaction pending flag should be reflected in the PFPCI_VMPEND register. The VM index is an absolute index in the range of 0 through 767. It can only be set by software to VMs that the PF owns and only to VMs that are not assigned to a VF.
RESERVED	31:10	0x0	RSV	N/A	Reserved.

13.2.2.3.30 PCIe VM Pending Status - PFPCI_VMPEND (0x0009E800; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PENDING	0	0b	RO	N/A	**PCIe Transaction Pending Status** The reported VM is controlled by the *VMINDEX* field in the PFPCI_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	N/A	Reserved.

13.2.2.3.31 Function Requester ID Information Register - PF_FUNC_RID (0x0009E880; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FUNCTION_NUMBER	2:0	000b	RO	N/A	**Function Number** Function number assigned to the function based on BIOS/OS enumeration.
DEVICE_NUMBER	7:3	0x0	RO	N/A	**Device Number** No-ARI mode: Device number assigned to the function based on BIOS/OS enumeration. ARI mode: Upper 5 bits of the 8-bit function number.
BUS_NUMBER	15:8	0x0	RO	N/A	**Bus Number** Bus number assigned to the function based on BIOS/OS enumeration.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.3.32 Function Active and Power State - PFPCI_FACTPS (0x0009E900; RO)

This register provides different indications about the function status.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FUNC_POWER_STATE	1:0	00b	RO	N/A	**Function Power State** Power state indication of the function. 00b = Dr 01b = D3 10b = D0a 11b = D0u
RESERVED	2	0b	RSV	N/A	Reserved.
FUNC_AUX_EN	3	0b	RO	N/A	**Function Aux Enable** Reflects the Auxiliary Power PM Enable bit from the PCI configuration space.
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.3.33 PCIe Statistic Control Register #5...#8 - GLPCI_GSCL_5_8[n] (0x0009E954 + 0x4*n, n=0...3; RW)

These registers control the operation of the leaky bucket counter n.

Field	Bit(s)	Init.	Type	CFG Policy	Description
LBC_THRESHOLD_N	15:0	0x0	RW	UNDEFINED	**Leaky Bucket Counter Threshold n** Threshold for the leaky bucket counter n.
LBC_TIMER_N	31:16	0x0	RW	UNDEFINED	**Leaky Bucket Counter Timer n** Time period between decrementing the value in leaky bucket Counter n. The time period is defined in μs units.

13.2.2.3.34 PCIe Byte Counter High - GLPCI_BYTCTH_P (0x0009E970; RO)

A byte counter used by the PCIe performance counters.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PCI_COUNT_BW_BCT	31:0	0x0	RO	N/A	**PCIe Byte Counter High** 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.

13.2.2.3.35 PCIe Byte Counter Low - GLPCI_BYTCTL_P (0x0009E994; RO)

A byte counter used by the PCIe performance counters.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PCI_COUNT_BW_BCT	31:0	0x0	RO	N/A	**PCIe Byte Counter Low** 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.

13.2.2.3.36 PCIe Statistic Control Registers #2 - GLPCI_GSCL_2 (0x0009E998; RW)

This register defines the events counted by the performance counters.

Field	Bit(s)	Init.	Type	CFG Policy	Description
GIO_EVENT_NUM_0	7:0	0x0	RW	UNDEFINED	**GIO Event Number 0** Event number that counter 0 counts (GSCN_0).
GIO_EVENT_NUM_1	15:8	0x0	RW	UNDEFINED	**GIO Event Number 1** Event number that counter 1 counts (GSCN_1).
GIO_EVENT_NUM_2	23:16	0x0	RW	UNDEFINED	**GIO Event Number 2** Event number that counter 2 counts (GSCN_2).
GIO_EVENT_NUM_3	31:24	0x0	RW	UNDEFINED	**GIO Event Number 3** Event number that counter 3 counts (GSCN_3).

13.2.2.3.37 PCIe Statistic Counter Registers #0...#3 - GLPCI_GSCN_0_3[n] (0x0009E99C + 0x4*n, n=0...3; RO)

These registers contain the performance counters 0-3.

Field	Bit(s)	Init.	Type	CFG Policy	Description
EVENT_COUNTER	31:0	0x0	RO	N/A	**Event Counter** A 32-bit event counter. See the section on Performance and Statistics Counters. These registers are stuck at their maximum value of 0xFF...F.

13.2.2.3.38 PCIe Default Revision ID - GLPCI_DREVID (0x0009E9AC; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DEFAULT_REVID	7:0	0x0	RO	N/A	**Default Revision ID** Mirroring of default Rev ID prior to an NVM load.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.3.39 PCIe Packet Counter - GLPCI_PKTCT_P (0x0009E9B0; RO)

A packet counter used by the PCIe performance counters.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PCI_COUNT_BW_PCT	31:0	0x0	RO	N/A	**PCIe Packet Counter** A 32-bit counter that counts PCIe packets. This register gets stuck at its maximum value of 0xFF...F.

13.2.2.3.40 PCIe Statistic Control Register #1 - GLPCI_GSCL_1_P (0x0009E9B4; RW)

This register controls the operation of the PCIe performance counters.

Field	Bit(s)	Init.	Type	CFG Policy	Description
GIO_COUNT_EN_0	0	0b	RW	UNDEFINED	**GIO Counter Enable 0** Enables PCIe statistic counter number 0.
GIO_COUNT_EN_1	1	0b	RW	UNDEFINED	**GIO Counter Enable 1** Enables PCIe statistic counter number 1.
GIO_COUNT_EN_2	2	0b	RW	UNDEFINED	**GIO Counter Enable 2** Enables PCIe statistic counter number 2.
GIO_COUNT_EN_3	3	0b	RW	UNDEFINED	**GIO Counter Enable 3** Enables PCIe statistic counter number 3.
LBC_ENABLE_0	4	0b	RW	UNDEFINED	**Leaky Bucket Counter Enable 0** 0b = Leaky bucket mode is disabled and the counter is incremented by one for each event. 1b = Statistics counter 0 operates in leaky bucket mode.
LBC_ENABLE_1	5	0b	RW	UNDEFINED	**Leaky Bucket Counter Enable 1** 0b = Leaky bucket mode is disabled and the counter is incremented by one for each event. 1b = Statistics counter 1 operates in leaky bucket mode.
LBC_ENABLE_2	6	0b	RW	UNDEFINED	**Leaky Bucket Counter Enable 2** 0b = Leaky bucket mode is disabled and the counter is incremented by one for each event. 1b = Statistics counter 2 operates in leaky bucket mode.
LBC_ENABLE_3	7	0b	RW	UNDEFINED	**Leaky Bucket Counter Enable 3** 0b = Leaky bucket mode is disabled and the counter is incremented by one for each event. 1b = Statistics counter 3 operates in leaky bucket mode.
RESERVED	27:8	0x0	RSV	N/A	Reserved.
GIO_64_BIT_EN	28	0b	RW	UNDEFINED	**GIO 64-bit Enable** Enables two 64-bit counters instead of four 32-bit counters.

Field	Bit(s)	Init.	Type	CFG Policy	Description
GIO_COUNT_RESET	29	0b	RW1S	DYNAMIC	**GIO Counters Reset** Reset indication of PCIe statistic counters. Reading this bit returns a 0b.
GIO_COUNT_STOP	30	0b	RW1S	DYNAMIC	**GIO Counters Stop** Stop indication of PCIe statistic counters. Reading this bit returns a 0b.
GIO_COUNT_START	31	0b	RW1S	DYNAMIC	**GIO Counters Start** Start indication of PCIe statistic counters. Reading this bit returns a 0b.

13.2.2.3.41 PCIe Global Config 2 - GLPCI_CNF2 (0x000BE004; RO)

This register contains global status fields of PCIe configuration.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RO_DIS	0	0b	RW	UNDEFINED	**Relaxed Ordering Disable** 0b = Relaxed ordering is specified per request type. 1b = The device does not request any relaxed ordering transactions.
CACHELINE_SIZE	1	0b	RW	UNDEFINED	**Cache Line Size** Determines the system cache line size. 0b = 64 bytes 1b = 128 bytes This field is loaded from the NVM.
RESERVED	31:2	0x0	RSV	N/A	Reserved.

13.2.2.3.42 PCIe Upper Address - GLPCI_UPADD (0x000BE0D4; RW)

This register is used to block PCIe master accesses above some address. See Section 3.1.5.7, "Blocking on Upper Address".

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	0	0b	RSV	N/A	Reserved.
ADDRESS	31:1	0x0	RW	UNDEFINED	**Address** Bits [31:1] correspond to bits [63:33] in the PCIe address space, respectively.

13.2.2.3.43 PCIe NPQ Config - GLPCI_NPQ_CFG (0x000BFD80; RW)

This register controls some parameters of NPQ.

Field	Bit(s)	Init.	Type	CFG Policy	Description
EXTEND_TO	0	0b	RW	UNDEFINED	**Extend Timeout** Extends timeout in the permitted range. 0b = Max 1b = Min
SMALL_TO	1	0b	RW	UNDEFINED	**Small Timeout** Reduce timeout value for simulation

Field	Bit(s)	Init.	Type	CFG Policy	Description
WEIGHT_AVG	5:2	0x2	RW	UNDEFINED	**Weight Average** Controls the weight of the average filter for average round trip calculation.
NPQ_SPARE	15:6	0x0	RW	UNDEFINED	**NPQ Spare Bits** Spare bits.
NPQ_ERR_STAT	19:16	0x0	RO	N/A	**NPQ Error Status**
RESERVED	31:20	0x0	RSV	N/A	Reserved.

13.2.2.3.44 PCIe NPQ Watermark of Pipe Monitor - GLPCI_WATMK_CLNT_PIPEMON (0x000BFD90; RO)

This register reports the current value of pipe monitor in NPQ.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DATA_LINES	15:0	0x4D9	RO	N/A	**Data Lines** Amount of data lines available.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.3.45 PCIe Packet Counter - GLPCI_PKTCT_NP_C (0x000BFD9C; RO)

A packet counter used by the PCIe performance counters.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PCI_COUNT_BW_PCT	31:0	0x0	RO	N/A	**PCIe Packet Counter** A 32-bit counter that counts PCIe packets. This register gets stuck at its maximum value of 0xFF...F.

13.2.2.3.46 PCIe Packet Counter - GLPCI_LATCT_NP_C (0x000BFDA0; RO)

A packet counter used by the PCIe performance counters.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PCI_LATENCY_COUNT	31:0	0x0	RO	N/A	**PCIe Latency Counter** When GPLCI_GSCL_0.*cfg_rt_event* = 0, maximal round trip. When GLPCI_GSCL_0.*cfg_rt_event* = 1, minimal round trip. When GLPCI_GSCL_0.*cfg_rt_event* = 2, average round trip.

13.2.2.3.47 PCIe Statistic Control Register #1 - GLPCI_GSCL_1_NP_C (0x000BFDA4; RW)

This register controls the operation of the PCIe performance counters.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	7:0	0x0	RSV	N/A	Reserved.
RT_MODE	8	0b	RW	UNDEFINED	**RT Mode**

Field	Bit(s)	Init.	Type	CFG Policy	Description
RT_EVENT	13:9	0x0	RW	UNDEFINED	**RT Event** 0x0 = Max 0x1 = Min 0x2 = Average
RESERVED	28:14	0x0	RSV	N/A	Reserved.
GIO_COUNT_RESET	29	0b	RW1S	DYNAMIC	**GIO Count Reset** Reset indication of PCIe statistic counters. Reading this bit returns a 0b.
GIO_COUNT_STOP	30	0b	RW1S	DYNAMIC	**GIO Count Stop** Stop indication of PCIe statistic counters. Reading this bit returns a 0b.
GIO_COUNT_START	31	0b	RW1S	DYNAMIC	**GIO Count Start** Start indication of PCIe statistic counters. Reading this bit returns a 0b.

13.2.2.3.48 PCIe Byte Counter High - GLPCI_BYTCTH_NP_C (0x000BFDA8; RO)

A byte counter used by the PCIe performance counters.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PCI_COUNT_BW_BCT	31:0	0x0	RO	N/A	**PCIe Byte Counter High** 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.

13.2.2.3.49 PCIe Byte Counter Low - GLPCI_BYTCTL_NP_C (0x000BFDAC; RO)

A byte counter used by the PCIe performance counters.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PCI_COUNT_BW_BCT	31:0	0x0	RO	N/A	**PCIe Byte Counter Low** 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.

13.2.2.4 PF - MAC Registers

13.2.2.4.1 HSEC CONTROL Receive PFC ENABLE - PRTMAC_HSEC_CTL_RX_PAUSE_ENABLE (0x001E3180; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
HSEC_CTL_RX_PAUSE_ENABLE	8:0	0x0	RW	UNDEFINED	**Rx Pause Enable** Rx priority flow control enable. This field is used to enable priority flow control per priority. When Bit[x] is set to 1b, PFC processing is enabled for priority-x Bit[8] is used to enable 802.3x flow control. ***Note:*** Priority flow control can be enabled only when the receive packet buffer of the port is configured for DCB mode.
RESERVED	31:9	0x0	RSV	N/A	Reserved.

13.2.2.4.2 HSEC CONTROL Transmit PAUSE_ENABLE - PRTMAC_HSEC_CTL_TX_PAUSE_ENABLE (0x001E31A0; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
HSEC_CTL_TX_PAUSE_ENABLE	8:0	0x0	RW	UNDEFINED	**Tx Pause Enable** Tx priority flow control enable. This field is used to enable priority flow control per priority. When Bit[x] is set to 1b, PFC packet transmission is enabled for Priority-X. Bit[8] is used to enable 802.3x flow control.
RESERVED	31:9	0x0	RSV	N/A	Reserved.

13.2.2.4.3 HSEC CONTROL Receive ENABLE_GCP - PRTMAC_HSEC_CTL_RX_ENABLE_GCP (0x001E31C0; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
HSEC_CTL_RX_ENABLE_GCP	0	1b	RW	UNDEFINED	**Rx Enable GCP** When set to 1b and *rx_forward_control* is set to 0, flow control packets are terminated by the HSEC MAC. When set to 0b, both 802.3x and PFC packet processing is disabled and the HSEC MAC forwards all control packets. ***Note:*** To enable 802.3x and PFC functional packet processing, a dedicated enable bit must be configured.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.4.4 HSEC CONTROL Receive PAUSE_DA_UCAST_PART1 - PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART1 (0x001E3220; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
HSEC_CTL_RX_PAUSE_DA_UCAST_PART1	31:0	0x0	RW	UNDEFINED	**Rx Pause Destination Address Unicast Part 1** Unicast destination address for pause processing. Valid only if the UC address is enabled for control processing through RX_CHECK_UC_PPP/PCP/GPP/GCP.

13.2.2.4.5 HSEC CONTROL Receive PAUSE_DA_UCAST_PART2 - PRTMAC_HSEC_CTL_RX_PAUSE_DA_UCAST_PART2 (0x001E3240; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
HSEC_CTL_RX_PAUSE_DA_UCAST_PART2	15:0	0x0	RW	UNDEFINED	**Rx Pause Destination Address Unicast Part 2** Unicast destination address for pause processing. Valid only if the UC address is enabled for control processing through RX_CHECK_UC_PPP/PCP/GPP/GCP.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.4.6 HSEC CONTROL Receive PAUSE_SA_PART1 - PRTMAC_HSEC_CTL_RX_PAUSE_SA_PART1 (0x001E3280; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
HSEC_CTL_RX_PAUSE_SA_PART1	31:0	0x0	RW	UNDEFINED	**Rx Pause Source Address Part 1** Source address for pause processing.

13.2.2.4.7 HSEC CONTROL Receive PAUSE_SA_PART2 - PRTMAC_HSEC_CTL_RX_PAUSE_SA_PART2 (0x001E32A0; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
HSEC_CTL_RX_PAUSE_SA_PART2	15:0	0x0	RW	UNDEFINED	**Rx Pause Source Address Part 2** Source address for pause processing.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.4.8 HSEC CONTROL Receive ENABLE_GPP - PRTMAC_HSEC_CTL_RX_ENABLE_GPP (0x001E34C0; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
HSEC_CTL_RX_ENABLE_GPP	0	0b	RW	UNDEFINED	**Rx Enable GPP** 1b = Enables 802.3x pause packet processing. ***Note:*** 802.3x pause is also referred to as Global Pause in HSEC registers.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.4.9 HSEC CONTROL Receive ENABLE_PPP - PRTMAC_HSEC_CTL_RX_ENABLE_PPP (0x001E35C0; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
HSEC_CTL_RX_ENABLE_PPP	0	0b	RW	UNDEFINED	**Rx Enable PPP** 1b = Enables priority pause packet processing.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.4.10 HSEC CONTROL Receive FORWARD_CONTROL - PRTMAC_HSEC_CTL_RX_FORWARD_CONTROL (0x001E36C0; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
HSEC_CTL_RX_FORWARD_CONTROL	0	0b	RW	UNDEFINED	**Rx Forward Control** 0b = Causes the HSEC MAC to drop control packets. 1b = Indicates that the HSEC MAC forwards control packets to the user.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.4.11 HSEC CONTROL Transmit PAUSE_QUANTA - PRTMAC_HSEC_CTL_TX_PAUSE_QUANTA[n] (0x001E36E0 + 0x20*n, n=0...8; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
HSEC_CTL_TX_PAUSE_QUANTA	15:0	0xFFFF	RW	UNDEFINED	**Tx Pause Quanta** These nine buses indicate the quanta to be transmitted for each of the eight priorities in priority-based pause operation and the global pause operation. The value for *stat_tx_pause_quanta*[8] is used for global pause operation. All other values are used for priority pause operation.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.4.12 HSEC CONTROL Transmit PAUSE_REFRESH_TIMER - PRTMAC_HSEC_CTL_TX_PAUSE_REFRESH_TIMER[n] (0x001E3800 + 0x20*n, n=0...8; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
HSEC_CTL_TX_PAUSE_REFRESH_TIMER	15:0	0x0	RW	UNDEFINED	**Tx Pause Refresh Timer** These nine buses set the retransmission time of pause packets for each of the eight priorities in priority-based pause operation and the global pause operation. The value for *stat_tx_pause_refresh_timer*[8] is used for global pause operation. All other values are used for priority pause operation.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.4.13 HSEC CONTROL Transmit SA_GPP_PART1 - PRTMAC_HSEC_CTL_TX_SA_PART1 (0x001E3960; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
HSEC_CTL_TX_SA_PART1	31:0	0x0	RW	UNDEFINED	**Tx Source Address Part 1** Source address for transmitting pause packets.

13.2.2.4.14 HSEC CONTROL Transmit SA_GPP_PART2 - PRTMAC_HSEC_CTL_TX_SA_PART2 (0x001E3980; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
HSEC_CTL_TX_SA_PART2	15:0	0x0	RW	UNDEFINED	**Tx Source Address Part 2** Source address for transmitting pause packets.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.4.15 MAC Rx Silent Drop Count - PRTMAC_RX_PKT_DRP_CNT (0x001E3C20; RW)

This register counts silent drop packets due to CORER, RX_ENABLE, and RPB overflow.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RX_PKT_DRP_CNT	15:0	0x0	RW1C	DYNAMIC	**Rx Packet Drop Count**
RX_MKR_PKT_DRP_CNT	31:16	0x0	RW1C	DYNAMIC	**Rx Marker Packet Drop Count**

13.2.2.4.16 MAC Rx Shift FC Quanta - PRTMAC_HSEC_CTL_RX_QUANTA_SHIFT (0x001E3C40; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRTMAC_HSEC_CTL_RX_QUANTA_SHIFT	15:0	0x0	RW	UNDEFINED	**Rx Quanta Shift** Shift left FC quanta counter values.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.4.17 MAC Rx Metadata Override Enable - PRTMAC_MD_OVRRIDE_ENABLE[n] (0x001E3C60 + 0x20*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRTMAC_MD_OVRRIDE_ENABLE	31:0	0x0	RW	UNDEFINED	**Metadata Override Enable** Debug feature to force value on metadata Rx.

13.2.2.4.18 MAC Rx Metadata Override Value - PRTMAC_MD_OVRRIDE_VAL[n] (0x001E3D60 +0x20*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRTMAC_MD_OVRRIDE_ENABLE	31:0	0x0	RW	UNDEFINED	**Metadata Override Enable** Debug feature to force value on metadata Rx.

13.2.2.4.19 Link Down Counter - PRTMAC_LINK_DOWN_COUNTER (0x001E47C0; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LINK_DOWN_COUNTER	15:0	0x0	RW1C	DYNAMIC	**Link Down Counter** Increments on link down event. Does not do wrap around.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.4.20 Link UP Counter Limit - PRTMAC_TX_LNK_UP_CNT (0x001E4840; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TX_LINK_UP_CNT	15:0	0x50	RW	UNDEFINED	**Tx Link Up Counter** Link UP counter limit.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.4.21 MAC Markers Counter Tx - PRTMAC_TX_CNT_MRKR (0x001E48C0; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TX_CNT_MRKR	15:0	0x0	RW1C	DYNAMIC	**Tx Count Marker**
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.4.22 MAC Markers Counter Rx - PRTMAC_RX_CNT_MRKR (0x001E48E0; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RX_CNT_MRKR	15:0	0x0	RW1C	DYNAMIC	**Rx Count Marker**
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.5 PF - Power Management Registers

Registers related to LTR and EEE.

13.2.2.5.1 PME_TO Indication - GLGEN_PME_TO (0x000B81BC; RO)

This register reflects an incoming PME turn-off message.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PME_TO_FOR_PE	0	0b	RO	N/A	**PME Turn-Off for PE** The bit is set to 1b when a *PME_Turn_off* message is received on the sideband IOSF and cleared upon PE core reset de-assertion.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.5.2 Global Power Mode Control S5 - GL_PWR_MODE_DIVIDE_S5_H_CTRL (0x000B81EC; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_50G_H	2:0	000b	RW	UNDEFINED	**Divide Value Total Bandwidth 50G** Core clock divide in S5 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 50G.
DIV_VAL_TBW_25G_H	5:3	001b	RW	UNDEFINED	**Divide Value Total Bandwidth 25G** Core clock divide in S5 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 25G.
DIV_VAL_TBW_10G_H	8:6	010b	RW	UNDEFINED	**Divide Value Total Bandwidth 10G** Core clock divide in S5 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 10G.
DIV_VAL_TBW_4G_H	11:9	011b	RW	UNDEFINED	**Divide Value Total Bandwidth 4G** Core clock divide in S5 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 4G.
DIV_VAL_TBW_A50G_H	15:12	0x0	RW	UNDEFINED	**Divide Value Total Bandwidth Above 50G** Core clock divide in S5 power mode when total bandwidth configuration is up to 100G, and total link speed is above 50G.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.5.3 Global Power Mode Control PE - GL_PWR_MODE_DIVIDE_S0_CTRL_H_PECLK (0x000B81F0; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_50G_H	2:0	000b	RW	UNDEFINED	**Divide Value Total Bandwidth 50G** PE core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 50G
DIV_VAL_TBW_25G_H	5:3	001b	RW	UNDEFINED	**Divide Value Total Bandwidth 25G** PE core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 25G.
DIV_VAL_TBW_10G_H	8:6	010b	RW	UNDEFINED	**Divide Value Total Bandwidth 10G** PE core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 10G.
DIV_VAL_TBW_4G_H	11:9	011b	RW	UNDEFINED	**Divide Value Total Bandwidth 4G** PE core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 4G.
DIV_VAL_TBW_A50G_H	15:12	0x0	RW	UNDEFINED	**Divide Value Total Bandwidth Above 50G** PE core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is above 50G.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.5.4 Global Power Mode Control Upper - GL_PWR_MODE_DIVIDE_S0_CTRL_H_UCLK (0x000B81F4; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_50G_H	2:0	000b	RW	UNDEFINED	**Divide Value Total Bandwidth 50G** Upper core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 50G.
DIV_VAL_TBW_25G_H	5:3	001b	RW	UNDEFINED	**Divide Value Total Bandwidth 25G** Upper core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 25G.
DIV_VAL_TBW_10G_H	8:6	010b	RW	UNDEFINED	**Divide Value Total Bandwidth 10G** Upper core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 10G.
DIV_VAL_TBW_4G_H	11:9	011b	RW	UNDEFINED	**Divide Value Total Bandwidth 4G** Upper core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 4G.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_A50G_H	15:12	0x0	RW	UNDEFINED	**Divide Value Total Bandwidth Above 50G** Upper core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is above 50G.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.5.5 Global Power Mode Control RXCTL - GL_PWR_MODE_DIVIDE_S0_CTRL_H_RXCTL (0x000B81F8; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_50G_H	2:0	000b	RW	UNDEFINED	**Divide Value Total Bandwidth 50G** RXCTL core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 50G.
DIV_VAL_TBW_25G_H	5:3	001b	RW	UNDEFINED	**Divide Value Total Bandwidth 25G** RXCTL core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 25G.
DIV_VAL_TBW_10G_H	8:6	010b	RW	UNDEFINED	**Divide Value Total Bandwidth 10G** RXCTL core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 10G.
DIV_VAL_TBW_4G_H	11:9	011b	RW	UNDEFINED	**Divide Value Total Bandwidth 4G** RXCTL core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 4G.
DIV_VAL_TBW_A50G_H	15:12	0x0	RW	UNDEFINED	**Divide Value Total Bandwidth Above 50G** RXCTL core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is above 50G.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.5.6 Global Power Mode Control PSM - GL_PWR_MODE_DIVIDE_S0_CTRL_H_PSM (0x000B81FC; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_50G_H	2:0	000b	RW	UNDEFINED	**Divide Value Total Bandwidth 50G** PSM core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 50G.
DIV_VAL_TBW_25G_H	5:3	001b	RW	UNDEFINED	**Divide Value Total Bandwidth 25G** PSM core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 25G.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_10G_H	8:6	010b	RW	UNDEFINED	**Divide Value Total Bandwidth 10G** PSM core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 10G.
DIV_VAL_TBW_4G_H	11:9	011b	RW	UNDEFINED	**Divide Value Total Bandwidth 4G** PSM core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 4G.
DIV_VAL_TBW_A50G_H	15:12	0x0	RW	UNDEFINED	**Divide Value Total Bandwidth Above 50G** PSM core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is above 50G.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.5.7 Global Power Mode Control Lower - GL_PWR_MODE_DIVIDE_S0_CTRL_H_LCLK (0x000B8200; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_50G_H	2:0	000b	RW	UNDEFINED	**Divide Value Total Bandwidth 50G** Lower core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 50G.
DIV_VAL_TBW_25G_H	5:3	001b	RW	UNDEFINED	**Divide Value Total Bandwidth 25G** Lower core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 25G.
DIV_VAL_TBW_10G_H	8:6	010b	RW	UNDEFINED	**Divide Value Total Bandwidth 10G** Lower core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 10G.
DIV_VAL_TBW_4G_H	11:9	011b	RW	UNDEFINED	**Divide Value Total Bandwidth 4G** Lower core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 4G.
DIV_VAL_TBW_A50G_H	15:12	0x0	RW	UNDEFINED	**Divide Value Total Bandwidth Above 50G** Lower core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is above 50G.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.5.8 Global Power Mode Control UANA - GL_PWR_MODE_DIVIDE_S0_CTRL_H_UANA (0x000B8208; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_50G_H	2:0	000b	RW	UNDEFINED	**Divide Value Total Bandwidth 50G** Upper ANA core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 50G.
DIV_VAL_TBW_25G_H	5:3	001b	RW	UNDEFINED	**Divide Value Total Bandwidth 25G** Upper ANA core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 25G.
DIV_VAL_TBW_10G_H	8:6	010b	RW	UNDEFINED	**Divide Value Total Bandwidth 10G** Upper ANA core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 10G.
DIV_VAL_TBW_4G_H	11:9	011b	RW	UNDEFINED	**Divide Value Total Bandwidth 4G** Upper ANA core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is up to 4G.
DIV_VAL_TBW_A50G_H	15:12	0x0	RW	UNDEFINED	**Divide Value Total Bandwidth Above 50G** Upper ANA core clock divide in S0 power mode when total bandwidth configuration is up to 100G, and total link speed is above 50G.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.5.9 Global Power Mode Control - GL_PWR_MODE_CTL (0x000B820C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SWITCH_PWR_MODE_EN	0	0b	RW	UNDEFINED	**Switch Power Mode Enable** Switch S0 power mode enable.
NIC_PWR_MODE_EN	1	0b	RW	UNDEFINED	**NIC Power Mode Enable** NIC S0 power mode enable.
S5_PWR_MODE_EN	2	1b	RW	UNDEFINED	**S5 Power Mode Enable** Global S5 power mode enable.
CAR_MAX_SW_CONFIG	4:3	00b	RW	UNDEFINED	**CAR Max Bandwidth Configuration** Max bandwidth fuse override. Can only decrease max bandwidth. 00b = 100 GbE 01b = 50 GbE 10b = 25 GbE 11b = 25 GbE
RESERVED	29:5	0x0	RSV	N/A	Reserved.
CAR_MAX_BW	31:30	00b	RO	N/A	**CAR Max Bandwidth** Final system max bandwidth. 00b = 200 GbE 01b = 100 GbE 10b = 50 GbE 11b = 25 GbE

13.2.2.5.10 Global Power Mode Control Defaults - GL_PWR_MODE_DIVIDE_CTRL_L_DEFAULT (0x000B8218; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DEFAULT_DIV_VAL_PECLK	2:0	001b	RW	UNDEFINED	**Default Divide Value PE Clock** Core clock divide with power mode disabled when total bandwidth is up to 25G.
DEFAULT_DIV_VAL_UCLK	5:3	001b	RW	UNDEFINED	**Default Divide Value Upper Clock** Core clock divide with power mode disabled when total bandwidth is up to 25G.
DEFAULT_DIV_VAL_LCLK	8:6	001b	RW	UNDEFINED	**Default Divide Value Lower Clock** Core clock divide with power mode disabled when total bandwidth is up to 25G.
DEFAULT_DIV_VAL_PSM	11:9	001b	RW	UNDEFINED	**Default Divide Value PSM** Core clock divide with power mode disabled when total bandwidth is up to 25G.
DEFAULT_DIV_VAL_RXCTL	14:12	001b	RW	UNDEFINED	**Default Divide Value RXCTL** Core clock divide with power mode disabled when total bandwidth is up to 25G.
DEFAULT_DIV_VAL_UANA	17:15	001b	RW	UNDEFINED	**Default Divide Value Upper ANA** Core clock divide with power mode disabled when total bandwidth is up to 25G.
DEFAULT_DIV_VAL_S5	20:18	001b	RW	UNDEFINED	**Default Divide Value S5** Core clock divide in S5 with power mode disabled when total bandwidth is up to 25G, and total link speed is up to 50G.
RESERVED	31:21	0x0	RSV	N/A	Reserved.

13.2.2.5.11 Global Power Mode Control PE - GL_PWR_MODE_DIVIDE_S0_CTRL_M_PECLK (0x000B821C; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_50G_M	2:0	000b	RW	UNDEFINED	**Divide Value Total Bandwidth 50G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 50G.
DIV_VAL_TBW_25G_M	5:3	001b	RW	UNDEFINED	**Divide Value Total Bandwidth 25G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 25G.
DIV_VAL_TBW_10G_M	8:6	010b	RW	UNDEFINED	**Divide Value Total Bandwidth 10G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 10G.
DIV_VAL_TBW_4G_M	11:9	011b	RW	UNDEFINED	**Divide Value Total Bandwidth 4G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 4G.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_A50G_M	14:12	000b	RW	UNDEFINED	**Divide Value Total Bandwidth Above 50G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is above 50G.
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.5.12 Global Power Mode Control PE - GL_PWR_MODE_DIVIDE_S0_CTRL_L_PECLK (0x000B8220; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_50G_L	2:0	000b	RW	UNDEFINED	**Divide Value Total Bandwidth 50G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 50G.
DIV_VAL_TBW_25G_L	5:3	001b	RW	UNDEFINED	**Divide Value Total Bandwidth 25G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 25G.
DIV_VAL_TBW_10G_L	8:6	010b	RW	UNDEFINED	**Divide Value Total Bandwidth 10G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 10G.
DIV_VAL_TBW_4G_L	11:9	011b	RW	UNDEFINED	**Divide Value Total Bandwidth 4G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 4G.
DIV_VAL_TBW_A50G_L	14:12	000b	RW	UNDEFINED	**Divide Value Total Bandwidth Above 50G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is above 50G.
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.5.13 Global Power Mode Control Upper - GL_PWR_MODE_DIVIDE_S0_CTRL_M_UCLK (0x000B8224; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_50G_M	2:0	000b	RW	UNDEFINED	**Divide Value Total Bandwidth 50G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 50G.
DIV_VAL_TBW_25G_M	5:3	001b	RW	UNDEFINED	**Divide Value Total Bandwidth 25G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 25G.
DIV_VAL_TBW_10G_M	8:6	010b	RW	UNDEFINED	**Divide Value Total Bandwidth 10G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 10G.
DIV_VAL_TBW_4G_M	11:9	011b	RW	UNDEFINED	**Divide Value Total Bandwidth 4G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 4G.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_A50G_M	14:12	000b	RW	UNDEFINED	**Divide Value Total Bandwidth Above 50G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is above 50G.
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.5.14 Global Power Mode Control RXCTL - GL_PWR_MODE_DIVIDE_S0_CTRL_M_RXCTL (0x000B8228; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_50G_M	2:0	000b	RW	UNDEFINED	**Divide Value Total Bandwidth 50G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 50G.
DIV_VAL_TBW_25G_M	5:3	001b	RW	UNDEFINED	**Divide Value Total Bandwidth 25G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 25G.
DIV_VAL_TBW_10G_M	8:6	010b	RW	UNDEFINED	**Divide Value Total Bandwidth 10G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 10G.
DIV_VAL_TBW_4G_M	11:9	011b	RW	UNDEFINED	**Divide Value Total Bandwidth 4G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 4G000b = no divide
DIV_VAL_TBW_A50G_M	14:12	000b	RW	UNDEFINED	**Divide Value Total Bandwidth Above 50G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is above 50G.
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.5.15 Global Power Mode Control PSM - GL_PWR_MODE_DIVIDE_S0_CTRL_M_PSM (0x000B822C; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_50G_M	2:0	000b	RW	UNDEFINED	**Divide Value Total Bandwidth 50G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 50G.
DIV_VAL_TBW_25G_M	5:3	001b	RW	UNDEFINED	**Divide Value Total Bandwidth 25G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 25G.
DIV_VAL_TBW_10G_M	8:6	010b	RW	UNDEFINED	**Divide Value Total Bandwidth 10G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 10G.
DIV_VAL_TBW_4G_M	11:9	011b	RW	UNDEFINED	**Divide Value Total Bandwidth 4G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 4G.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_A50G_M	14:12	000b	RW	UNDEFINED	**Divide Value Total Bandwidth Above 50G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is above 50G.
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.5.16 Global Power Mode Control Lower - GL_PWR_MODE_DIVIDE_S0_CTRL_M_LCLK (0x000B8230; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_50G_M	2:0	000b	RW	UNDEFINED	**Divide Value Total Bandwidth 50G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 50G.
DIV_VAL_TBW_25G_M	5:3	001b	RW	UNDEFINED	**Divide Value Total Bandwidth 25G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 25G.
DIV_VAL_TBW_10G_M	8:6	010b	RW	UNDEFINED	**Divide Value Total Bandwidth 10G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 10G.
DIV_VAL_TBW_4G_M	11:9	011b	RW	UNDEFINED	**Divide Value Total Bandwidth 4G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 4G.
DIV_VAL_TBW_A50G_M	14:12	000b	RW	UNDEFINED	**Divide Value Total Bandwidth Above 50G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is above 50G.
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.5.17 Global Power Mode Control UANA - GL_PWR_MODE_DIVIDE_S0_CTRL_M_UANA (0x000B8234; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_50G_M	2:0	000b	RW	UNDEFINED	**Divide Value Total Bandwidth 50G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 50G.
DIV_VAL_TBW_25G_M	5:3	001b	RW	UNDEFINED	**Divide Value Total Bandwidth 25G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 25G.
DIV_VAL_TBW_10G_M	8:6	010b	RW	UNDEFINED	**Divide Value Total Bandwidth 10G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 10G.
DIV_VAL_TBW_4G_M	11:9	011b	RW	UNDEFINED	**Divide Value Total Bandwidth 4G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is up to 4G.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_A50G_M	14:12	000b	RW	UNDEFINED	**Divide Value Total Bandwidth Above 50G** Core clock divide in S0 power mode when total bandwidth is up to 50G, and total link speed is above 50G.
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.5.18 Global Power Mode Control Upper - GL_PWR_MODE_DIVIDE_S0_CTRL_L_UCLK (0x000B8238; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_50G_L	2:0	000b	RW	UNDEFINED	**Divide Value Total Bandwidth 50G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 50G.
DIV_VAL_TBW_25G_L	5:3	001b	RW	UNDEFINED	**Divide Value Total Bandwidth 25G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 25G.
DIV_VAL_TBW_10G_L	8:6	010b	RW	UNDEFINED	**Divide Value Total Bandwidth 10G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 10G.
DIV_VAL_TBW_4G_L	11:9	011b	RW	UNDEFINED	**Divide Value Total Bandwidth 4G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 4G.
DIV_VAL_TBW_A50G_L	14:12	000b	RW	UNDEFINED	**Divide Value Total Bandwidth Above 50G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is above 50G.
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.5.19 Global Power Mode Control RXCTL - GL_PWR_MODE_DIVIDE_S0_CTRL_L_RXCTL (0x000B823C; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_50G_L	2:0	000b	RW	UNDEFINED	**Divide Value Total Bandwidth 50G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 50G.
DIV_VAL_TBW_25G_L	5:3	001b	RW	UNDEFINED	**Divide Value Total Bandwidth 25G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 25G.
DIV_VAL_TBW_10G_L	8:6	010b	RW	UNDEFINED	**Divide Value Total Bandwidth 10G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 10G.
DIV_VAL_TBW_4G_L	11:9	011b	RW	UNDEFINED	**Divide Value Total Bandwidth 4G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 4G.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_A50G_L	14:12	000b	RW	UNDEFINED	**Divide Value Total Bandwidth Above 50G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is above 50G.
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.5.20 Global Power Mode Control PSM - GL_PWR_MODE_DIVIDE_S0_CTRL_L_PSM (0x000B8240; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_50G_L	2:0	000b	RW	UNDEFINED	**Divide Value Total Bandwidth 50G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 50G.
DIV_VAL_TBW_25G_L	5:3	001b	RW	UNDEFINED	**Divide Value Total Bandwidth 25G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 25G.
DIV_VAL_TBW_10G_L	8:6	010b	RW	UNDEFINED	**Divide Value Total Bandwidth 10G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 10G.
DIV_VAL_TBW_4G_L	11:9	011b	RW	UNDEFINED	**Divide Value Total Bandwidth 4G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 4G.
DIV_VAL_TBW_A50G_L	14:12	000b	RW	UNDEFINED	**Divide Value Total Bandwidth Above 50G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is above 50G.
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.5.21 Global Power Mode Control Lower - GL_PWR_MODE_DIVIDE_S0_CTRL_L_LCLK (0x000B8244; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_50G_L	2:0	000b	RW	UNDEFINED	**Divide Value Total Bandwidth 50G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 50G.
DIV_VAL_TBW_25G_L	5:3	001b	RW	UNDEFINED	**Divide Value Total Bandwidth 25G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 25G.
DIV_VAL_TBW_10G_L	8:6	010b	RW	UNDEFINED	**Divide Value Total Bandwidth 10G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 10G.
DIV_VAL_TBW_4G_L	11:9	011b	RW	UNDEFINED	**Divide Value Total Bandwidth 4G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 4G.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_A50G_L	14:12	000b	RW	UNDEFINED	**Divide Value Total Bandwidth Above 50G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is above 50G.
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.5.22 Global Power Mode Control UANA - GL_PWR_MODE_DIVIDE_S0_CTRL_L_UANA (0x000B8248; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_50G_L	2:0	000b	RW	UNDEFINED	**Divide Value Total Bandwidth 50G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 50G.
DIV_VAL_TBW_25G_L	5:3	001b	RW	UNDEFINED	**Divide Value Total Bandwidth 25G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 25G.
DIV_VAL_TBW_10G_L	8:6	010b	RW	UNDEFINED	**Divide Value Total Bandwidth 10G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 10G.
DIV_VAL_TBW_4G_L	11:9	011b	RW	UNDEFINED	**Divide Value Total Bandwidth 4G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is up to 4G.
DIV_VAL_TBW_A50G_L	14:12	000b	RW	UNDEFINED	**Divide Value Total Bandwidth Above 50G** Core clock divide in S0 power mode when total bandwidth is up to 25G, and total link speed is above 50G.
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.5.23 Global Power Mode Control S5 - GL_PWR_MODE_DIVIDE_S5_L_CTRL (0x000B824C; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_50G_L	2:0	000b	RW	UNDEFINED	**Divide Value Total Bandwidth 50G** Core clock divide in S5 power mode when total bandwidth is up to 25G, and total link speed is up to 50G.
DIV_VAL_TBW_25G_L	5:3	001b	RW	UNDEFINED	**Divide Value Total Bandwidth 25G** Core clock divide in S5 power mode when total bandwidth is up to 25G, and total link speed is up to 25G.
DIV_VAL_TBW_10G_L	8:6	010b	RW	UNDEFINED	**Divide Value Total Bandwidth 10G** Core clock divide in S5 power mode when total bandwidth is up to 25G, and total link speed is up to 10G.
DIV_VAL_TBW_4G_L	11:9	011b	RW	UNDEFINED	**Divide Value Total Bandwidth 4G** Core clock divide in S5 power mode when total bandwidth is up to 25G, and total link speed is up to 4G.
DIV_VAL_TBW_A50G_L	14:12	000b	RW	UNDEFINED	**Divide Value Total Bandwidth Above 50G** Core clock divide in S5 power mode when total bandwidth is up to 25G, and total link speed is above 50G.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.5.24 Global Power Mode Control S5 - GL_PWR_MODE_DIVIDE_S5_M_CTRL (0x000B8250; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DIV_VAL_TBW_50G_M	2:0	000b	RW	UNDEFINED	**Divide Value Total Bandwidth 50G** Core clock divide in S5 power mode when total bandwidth is up to 50G, and total link speed is up to 50G.
DIV_VAL_TBW_25G_M	5:3	001b	RW	UNDEFINED	**Divide Value Total Bandwidth 25G** Core clock divide in S5 power mode when total bandwidth is up to 50G, and total link speed is up to 25G.
DIV_VAL_TBW_10G_M	8:6	010b	RW	UNDEFINED	**Divide Value Total Bandwidth 10G** Core clock divide in S5 power mode when total bandwidth is up to 50G, and total link speed is up to 10G.
DIV_VAL_TBW_4G_M	11:9	011b	RW	UNDEFINED	**Divide Value Total Bandwidth 4G** Core clock divide in S5 power mode when total bandwidth is up to 50G, and total link speed is up to 4G.
DIV_VAL_TBW_A50G_M	14:12	000b	RW	UNDEFINED	**Divide Value Total Bandwidth Above 50G** Core clock divide in S5 power mode when total bandwidth is up to 50G, and total link speed is above 50G.
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.5.25 Global Power Mode Control Defaults - GL_PWR_MODE_DIVIDE_CTRL_H_DEFAULT (0x000B825C; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DEFAULT_DIV_VAL_PECLK	2:0	000b	RW	UNDEFINED	**Default Divide Value PE Clock** Core clock divide with power mode disabled when total bandwidth is up to 100G.
DEFAULT_DIV_VAL_UCLK	5:3	000b	RW	UNDEFINED	**Default Divide Value Upper Clock** Core clock divide with power mode disabled when total bandwidth is up to 100G.
DEFAULT_DIV_VAL_LCLK	8:6	000b	RW	UNDEFINED	**Default Divide Value Lower Clock** Core clock divide with power mode disabled when total bandwidth is up to 100G.
DEFAULT_DIV_VAL_PSM	11:9	000b	RW	UNDEFINED	**Default Divide Value PSM** Core clock divide with power mode disabled when total bandwidth is up to 100G.
DEFAULT_DIV_VAL_RXCTL	14:12	000b	RW	UNDEFINED	**Default Divide Value RXCTL** Core clock divide with power mode disabled when total bandwidth is up to 100G.
DEFAULT_DIV_VAL_UANA	17:15	000b	RW	UNDEFINED	**Default Divide Value Upper ANA** Core clock divide with power mode disabled when total bandwidth is up to 100G.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DEFAULT_DIV_VAL_S5	20:18	000b	RW	UNDEFINED	**Default Divide Value S5** Core clock divide with power mode disabled when total bandwidth is up to 100G.
RESERVED	31:21	0x0	RSV	N/A	Reserved.

13.2.2.5.26 Global Power Mode Control Defaults - GL_PWR_MODE_DIVIDE_CTRL_M_DEFAULT (0x000B8260; RO)

000b = No divide; 001b = Divide by 2; 010b = Divide by 4; 011b = Divide by 8; 100b = Divide by 16. All other values are reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DEFAULT_DIV_VAL_PECLK	2:0	000b	RW	UNDEFINED	**Default Divide Value PE Clock** Core clock divide with power mode disabled when total bandwidth is up to 50G.
DEFAULT_DIV_VAL_UCLK	5:3	000b	RW	UNDEFINED	**Default Divide Value Upper Clock** Core clock divide with power mode disabled when total bandwidth is up to 50G.
DEFAULT_DIV_VAL_LCLK	8:6	000b	RW	UNDEFINED	**Default Divide Value Lower Clock** Core clock divide with power mode disabled when total bandwidth is up to 50G.
DEFAULT_DIV_VAL_PSM	11:9	000b	RW	UNDEFINED	**Default Divide Value PSM** Core clock divide with power mode disabled when total bandwidth is up to 50G.
DEFAULT_DIV_VAL_RXCTL	14:12	000b	RW	UNDEFINED	**Default Divide Value RXCTL** Core clock divide with power mode disabled when total bandwidth is up to 50G.
DEFAULT_DIV_VAL_UANA	17:15	000b	RW	UNDEFINED	**Default Divide Value Upper ANA** Core clock divide with power mode disabled when total bandwidth is up to 50G.
DEFAULT_DIV_VAL_S5	20:18	000b	RW	UNDEFINED	**Default Divide Value S5** Core clock divide with power mode disabled when total bandwidth is up to 50G.
RESERVED	31:21	0x0	RSV	N/A	Reserved.

13.2.2.5.27 Global Power Mode Control - GL_S5_PWR_MODE_EXIT_CTL (0x000B8270; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
S5_PWR_MODE_AUTO_EXIT	0	1b	RW	UNDEFINED	**S5 Power Mode Auto Exit** When set, hardware exits S5 power mode automatically when moving to S0.
RESERVED	2:1	00b	RSV	N/A	Reserved.
S5_PWR_MODE_PRST_FLOWS_ON_CORER	3	1b	RW	UNDEFINED	**S5 Power Mode PRST Flows on CORER**
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.5.28 Energy Efficient Ethernet (EEE) Status - PRTPM_EEE_STAT (0x001E4320; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	28:0	0x0	RSV	N/A	Reserved. Write 0. Ignore on read.
EEE_NEG	29	0b	RO	N/A	**EEE Negotiated** EEE support negotiated on link. 0b = EEE operation not supported on link. 1b = EEE operation supported on link.
RX_LPI_STATUS	30	0b	RO	N/A	**Rx LPI Status** Rx link in LPI state. 0b = Rx in Active state. 1b = Rx in LPI state.
TX_LPI_STATUS	31	0b	RO	N/A	**Tx LPI Status** Tx link in LPI state. 0b = Tx in Active state. 1b = Tx in LPI state.

13.2.2.5.29 Energy Efficient Ethernet (EEE) Register - PRTPM_EEER (0x001E4360; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TW_SYSTEM	15:0	0x0	RW	UNDEFINED	**TW System** 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. ***Notes:*** 1. 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*. 2. Following link disconnect or auto-negotiation, value of this field returns to default value, until software re-negotiates new *Tw_sys_tx* value via EEE LLDP. 3. Fast Retrain and Local/Remote Fault indication are not considered link disconnect, and do not cause the field to return to the default value. 4. 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	UNDEFINED	**TW LPI Enable** 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:*** 1. Even when *TX_LPI_EN* is 1b, device will not enable entry into Tx LPI state for at least PRTPM_EEEC.*TX_LU_LPI_DLY* following the change of *link_status* to OK as defined in IEEE802.3az clause 78.1.2.1. 2. 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 or forced by software to enable EEE on non AN protocols.
RESERVED	31:17	0x0	RSV	N/A	Reserved. Write 0. Ignore on read.

13.2.2.5.30 Energy Efficient Ethernet (EEE) Control - PRTPM_EEEC (0x001E4380; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	15:0	0x0	RSV	N/A	Reserved. Write 0. Ignore on read.
TW_WAKE_MIN	21:16	0xA	RW	UNDEFINED	**TW Wake Minimum** 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. ***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	00b	RSV	N/A	Reserved.
TX_LU_LPI_DLY	25:24	011b	RW	UNDEFINED	**Tx Link-Up LPI Delay** 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.
TEEE_DLY	31:26	0x3	RW	UNDEFINED	**Tx EEE Delay** Tx EEE LPI entry delay. Field defines delay to EEE entry once conditions to enter EEE LPI are detected. Field resolution is 1 µs. ***Notes:*** 1. If conditions to enter LPI during the *TEEE_DLY* interval cease to exist, device will not enter Tx LPI and continues normal operation. 2. Minimum configuration should be 0x1.

13.2.2.5.31 EEE Rx LPI Count - PRTPM_RLPIC (0x001E43A0; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
ERLPIC	31:0	0x0	RO	N/A	**EEE Rx LPI Counter** Counts EEE Rx LPI entry events. A EEE Rx LPI event occurs when the receiver detects link partner entry into EEE (IEEE802.3az) LPI state. This register only increments if receives are enabled and EEE operation is enabled. Register is cleared on read.

13.2.2.5.32 EEE Tx LPI Count - PRTPM_TLPIC (0x001E43C0; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
ETLPIC	31:0	0x0	RO	N/A	**EEE Tx LPI Counter** Counts EEE Tx LPI entry events. A EEE Tx LPI event occurs when the transmitter enters EEE (IEEE802.3az) LPI state. This register only increments if transmits are enabled and EEE operation is enabled. Register is cleared on read.

13.2.2.5.33 EEE Tx Control - PRTPM_EEETXC (0x001E43E0; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TW_PHY	15:0	0x1	RW	UNDEFINED	**TW PHY** *TW_PHY* value is set by firmware and should accommodate for the time defined in IEEE802.3az for the per connected PHY technology *TW_PHY* with the addition of the proper per-PHY technology addition as defined in the GLPM_EEE_SU and GLPM_EEE_SU_EXT registers. Value defined in this field is expressed in 102.4 nanosecond resolution. ***Note:*** The idle time value defined by this field 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	31:16	0x0	RSV	N/A	Reserved.

13.2.2.5.34 EEE Tx FW Done - PRTPM_EEEFWD (0x001E4400; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	30:0	0x0	RSV	N/A	Reserved.
EEE_FW_CONFIG_DONE	31	0b	RW	UNDEFINED	**EEE Firmware Configuration Done** Set by firmware to indicate that firmware configuration of the EEE parameters after link establishment is done. Cleared by hardware when link is down.

13.2.2.6 PF - Wake-Up Registers

13.2.2.6.1 Wake-Up Status Register - PFPM_WUS (0x0009DB80; RW1C)

This register is used to record statistics about all wake-up packets received. If a packet matches multiple criteria, then multiple bits could be set. Writing a 1b to any bit clears that bit.

This register is not cleared when PE_RST_N is asserted (excepted for *PME_STATUS* bit). It is only cleared when LAN_PWR_GOOD is de-asserted or when cleared by the software device driver.

Note: If additional packets are received that match one of the wake-up filters, after the original wake-up packet is received, the WUS register is not updated with the new match detection until the register is cleared.

Wakeup statuses are reported in this register also when no wake-up event is generated because a CRC error is detected on the presumed wake-up packet.

Field	Bit(s)	Init.	Type	CFG Policy	Description
LNKC	0	0b	RW1C	DYNAMIC	**Link Change** Link status changed.
MAG	1	0b	RW1C	DYNAMIC	**Magic** Magic packet received.
PME_STATUS	2	0b	RW1C	DYNAMIC	**PME Status** This bit is set when 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 also clears the *PME_Status* bit in the PMCSR. Bit is reset only on power-on reset (LAN_PWR_GOOD). When AUX_PWR = 0, bit is reset also on de-assertion of PE_RST_N.
MNG	3	0b	RW1C	DYNAMIC	**MNG** MNG wake-up status.
RESERVED	30:4	0x0	RSV	N/A	Reserved.
FW_RST_WK	31	0b	RW1C	DYNAMIC	**Firmware Reset Wake** Wake due to firmware reset assertion event. When set to 1b, indicates that firmware reset assertion caused system wake so that software driver can re-send proxying information to firmware.

13.2.2.6.2 Wake-Up Filter Control Register - PFPM_WUFC (0x0009DC00; RW)

This register is used to enable each of the pre-defined filters for wake-up support. A value of 1b means the filter is turned on, a value of 0b means the filter is turned off.

The bits reset only on power-on reset (LAN_PWR_GOOD) and on D3/Dr to D0 transitions.

Field	Bit(s)	Init.	Type	CFG Policy	Description
LNKC	0	0b	RW	UNDEFINED	**Link Change** Link status change wake-up enable.
MAG	1	0b	RW	UNDEFINED	**Magic** Magic packet wake-up enable.
RESERVED	2	0b	RSV	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MNG	3	0b	RW	UNDEFINED	**MNG** MNG wake-up enable.
RESERVED	30:4	0x0	RSV	N/A	Reserved.
FW_RST_WK	31	0b	RW	UNDEFINED	**Firmware Reset Wake** Enable wake on firmware reset assertion. When set a firmware reset causes system wake so that software driver can re-send Proxying information to firmware.

13.2.2.6.3 Wake-Up Control Register - PFPM_WUC (0x0009DC80; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	4:0	0x0	RSV	N/A	Reserved.
EN_APM_D0	5	0b	RW	UNDEFINED	**Enable APM D0** Enable APM wake also on D0. 0b = Enable wake only when function is in D3/Dr. 1b = Enable wake also in D0. Should be set for normal operation.
RESERVED	31:6	0x0	RSV	N/A	Reserved.

13.2.2.6.4 Wake-Up on MNG Control - GLPM_WUMC (0x0009DEE4; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	15:0	0x0	RSV	N/A	Reserved.
MNG_WU_PF	23:16	0x0	RW	UNDEFINED	**MNG Wake-Up PF** *MNG_WU_PF* EMP can set a bit in this field to indicate MNG initiated wake-up event (bit per PF).
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.6.5 APM Control Register - PFPM_APM (0x000B8080; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
APME	0	0b	RW	UNDEFINED	**Advance Power Management Enable** If set to 1b, APM wake-up is enabled. ***Note:*** Bit is reset on Power on reset (LAN_PWR_GOOD) only.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.6.6 MAC Address Low - PRTPM_SAL[n] (0x001E3B20 + 0x20*n, n=0...3; RW)

This register contains the station address lower 32 bits of the 48-bit Ethernet MAC Address. It is loaded by firmware from NVM and can be altered later by PF driver admin command.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PFPM_SAL	31:0	0x0	RW	UNDEFINED	**Station Address Low** The lower 32 bits of the 48-bit NVM pre-assigned Ethernet MAC Address. ***Note:*** Field is defined in Big Endian (LS byte of SAL is first on the wire).

13.2.2.6.7 MAC Address High - PRTPM_SAH[n] (0x001E3BA0 + 0x20*n, n=0...3; RW)

This register contains the station address upper 16 bits of the 48-bit Ethernet MAC Address. It is loaded by firmware from NVM and can be altered later by PF driver admin command.

PRTPM_SAH.*AV* determines whether this address is valid and compared against the incoming packet. After reset, firmware loads the relevant MAC Address from NVM, and sets its *Address Valid* field to 1b.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PFPM_SAH	15:0	0x0	RW	UNDEFINED	**Station Address High** The upper 16 bits of the 48-bit Ethernet MAC Address. ***Note:*** Field is defined in Big Endian (MS byte of *PRTPM_SAH* is last on the wire).
RESERVED	25:16	0x0	RSV	N/A	Reserved.
PF_NUM	29:26	0x0	RW	UNDEFINED	**PF Number** PF number to be used for reporting the waking PF. Value is written by firmware.
MC_MAG_EN	30	0b	RW	UNDEFINED	**Multicast Magic Packet Enable** Enable promiscuous multicast for Magic Packets. If this bit is set to 1b, every multicast Magic Packet generates a WoL event if enabled in PFPM_WUFC.*MAG*
AV	31	0b	RW	UNDEFINED	**Address Valid** If the NVM is present, the station address is assigned by firmware after loading from the NVM and its *Address Valid* field is set to 1b

13.2.2.7 PF - NVM Registers

13.2.2.7.1 HLP Auto-Load Done Register - GLNVM_AL_DONE_HLP (0x000824C4; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
HLP_CORER	0	0b	RO	N/A	**HLP Core Reset** Auto-load done indication.
HLP_FULLR	1	0b	RO	N/A	**HLP Full Reset** Auto-load done indication.
RESERVED	31:2	0x0	RSV	N/A	**Reserved.**

13.2.2.7.2 Unit Load Status - GLNVM_ULD (0x000B6008; RO)

This register provides indications on the completion of loading the Shadow RAM and Alternate Module into the device units.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PCIER_DONE	0	0b	RO	N/A	**PCIe Reset Done** The PCIe Reset process is done (all related registers are loaded).
PCIER_DONE_1	1	0b	RO	N/A	**PCIe Reset Done 1** The PCIe Reset process is done (all related registers are loaded). Mirror of Bit 0.
RESERVED	2	1b	RSV	N/A	Reserved.
CORER_DONE	3	0b	RO	N/A	**Core Reset Done** The Core Reset process is done (all related registers are loaded).
GLOBR_DONE	4	0b	RO	N/A	**Global Reset Done** The Global Reset process is done (all related registers are loaded).
POR_DONE	5	0b	RO	N/A	**Power On Reset Done** The Power On Reset process is done (all related registers are loaded).
RESERVED	7:6	11b	RSV	N/A	Reserved.
POR_DONE_1	8	0b	RO	N/A	**Power On Reset Done 1** The Power On Reset process is done (all related registers are loaded). Mirror of Bit 5.
PCIER_DONE_2	9	0b	RO	N/A	**PCIe Reset Done 2** The PCIe Reset process is done (all related registers are loaded). Mirror of bit 0.
PE_DONE	10	0b	RO	N/A	**Protocol Engine Done** The Protocol Engine Core Reset process is done (all related registers are loaded).
RESERVED	31:11	0x0	RSV	N/A	Reserved.

13.2.2.7.3 Protected CSR List - GLNVM_PROTCSR[n] (0x000B6010 + 0x4*n, n=0...59; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
ADDR_BLOCK	23:0	0xFFFFFF	RW	UNDEFINED	**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	N/A	Reserved.

13.2.2.7.4 Global NVM General Status Register - GLNVM_GENS (0x000B6100; RO)

This register cannot be loaded from NVM via one of the CSR format modules.

Field	Bit(s)	Init.	Type	CFG Policy	Description
NVM_PRES	0	0b	RW	N/A	**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	N/A	Reserved.
SR_SIZE	7:5	110b	RO	N/A	**Shadow RAM Size** This field defines the size of the internal Shadow RAM. The Shadow RAM size is equal to 2^*SR_Size* KB. Initial value is 110b, which corresponds to 2^6 = 64 KB.
BANK1VAL	8	0b	RW	N/A	**Basic Bank 1 Valid** 0b = Indicates that the content of basic banks 0 of the Flash device is valid. 1b = Indicates that the content of basic bank 1 of the Flash device 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	N/A	Reserved.

13.2.2.7.5 Flash Access Register - GLNVM_FLA (0x000B6108; RO)

This register is writable by the host only when the device is in the blank Flash programming mode. It cannot be loaded from NVM via one of the CSR format modules.

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.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	5:0	0x0	RSV	N/A	Reserved.
LOCKED	6	1b	RW	UNDEFINED	**Locked** Normal NVM programming mode. 0b = The device is in the blank Flash programming mode. 1b = The device is in the normal NVM programming mode. The bit can be cleared by EMP or by software.
RESERVED	31:7	0x0	RSV	N/A	Reserved.

13.2.2.7.6 Auto-Load Timers - GLNVM_ALTIMERS (0x000B6140; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PCI_ALTIMER	11:0	0x010	RW	UNDEFINED	**PCIe Auto-Load Timer** Default is 16 to 15 ms. Resolution is in ms.
GEN_ALTIMER	31:12	0x0012C	RW	UNDEFINED	**General Auto-Load Timer** Default is ~ 300 ms. Resolution is in ms.

13.2.2.7.7 Unit Load Timeout - GLNVM_ULT (0x000B6154; RO)

This register cannot be loaded from NVM via one of the CSR format modules.

Field	Bit(s)	Init.	Type	CFG Policy	Description
CONF_PCIR_AE	0	0b	RW	UNDEFINED	When set, indicates that the PCIe Auto-load Timer has ended before the respective NVM module has been initialized.
CONF_PCIRTL_AE	1	0b	RW	UNDEFINED	When set, indicates that the PCIe Auto-load Timer has ended before the respective NVM module has been initialized.
RESERVED	2	0b	RSV	N/A	Reserved.
CONF_CORE_AE	3	0b	RW	UNDEFINED	When set, indicates that the General Auto-load Timer has ended before the respective NVM module has been initialized.
CONF_GLOBAL_AE	4	0b	RW	UNDEFINED	When set, indicates that the General Auto-load Timer has ended before the respective NVM module has been initialized.
CONF_POR_AE	5	0b	RW	UNDEFINED	When set, indicates that the General Auto-load Timer has ended before the respective NVM module has been initialized.
RESERVED	8:6	000b	RSV	N/A	Reserved.
CONF_PCIALT_AE	9	0b	RW	UNDEFINED	When set, indicates that the PCIe Auto-load Timer has ended before the respective NVM module has been initialized.
CONF_PE_AE	10	0b	RW	UNDEFINED	When set, indicates that the PCIe Auto-load Timer has ended before the respective NVM module has been initialized.
RESERVED	31:11	0x0	RSV	N/A	Reserved.

13.2.2.8 PF - Analyzer Registers (Pre Parser)

Registers used for the pre-parser analyzer configuration.

13.2.2.8.1 L2 Tag Data Low - GL_SWT_L2TAG0[n] (0x000492A8 + 0x4*n, n=0...7; RO)

The data to insert in the fixed part of the L2 tag. Which bytes to take is fixed by the L2TAGTXIB register.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DATA	31:0	0x0	RW	UNDEFINED	**Data** L2TAG Data bytes. Fixed bytes 0-3 of the L2 tag. Unused part of these bytes and the word in which the variable part is inserted should be set to zero.

13.2.2.8.2 L2 Tag Data High - GL_SWT_L2TAG1[n] (0x000492C8 + 0x4*n, n=0...7; RO)

The data to insert in the fixed part of the L2 tag. Which bytes to take is fixed by the L2TAGTXIB register.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DATA	31:0	0x0	RW	UNDEFINED	**Data** L2TAG Data bytes. Fixed bytes 4-7 of the L2 tag. Unused part of these bytes and the word in which the variable part is inserted should be set to zero.

13.2.2.8.3 L2 Tag Tx Insert Bytes - GL_SWT_L2TAGTXIB[n] (0x000492E8 + 0x4*n, n=0...7; RO)

This register describes where to insert the variable part for the tags.

Field	Bit(s)	Init.	Type	CFG Policy	Description
OFFSET	7:0	0x0	RW	UNDEFINED	**Offset** Describes the offset (in bytes) in the header to which the variable data should be inserted.
LENGTH	9:8	00b	RW	UNDEFINED	**Length** Defines the length of the tag to insert. 00b = 8 bits 01b = 16 bits 10b = 24 bits 11b = 32 bits
RESERVED	31:10	0x0	RSV	N/A	Reserved.

13.2.2.8.4 L2 Tag Rx Extract Bytes - GL_SWT_L2TAGRXEB[n] (0x00052000 + 0x4*n, n=0...7; RW)

Rx byte enable for L2 tag.

Field	Bit(s)	Init.	Type	CFG Policy	Description
OFFSET	7:0	0x0	RW	UNDEFINED	**Offset** Describes the offset in the header from which the variable data should be extracted. Offset is given in bytes and does not include the EtherType.
LENGTH	9:8	00b	RW	UNDEFINED	**Length** Variable part length. 00b = 8 bits 01b = 16 bits 10b = 24 bits 11b = 32 bits
RESERVED	31:10	0x0	RSV	N/A	Reserved.

13.2.2.8.5 L2 Tag Control - GL_SWT_L2TAGCTRL[n] (0x001D2660 + 0x4*n, n=0...7; RW)

Control register for L2 tags.

Field	Bit(s)	Init.	Type	CFG Policy	Description
LENGTH	6:0	0x0	RW	UNDEFINED	**Length** Describes the number of bytes expected for this tag, not including the EtherType (2/4/6/8 if *LONG* bit is not set, any value otherwise). Length is in bytes (up to 126).
HAS_UP	7	0b	RW	UNDEFINED	**Has UP** Indicates that this tag has a UP field, and this field should be taken into account for UP-to-TC translation if this is the first tag of the packet.
RESERVED	8	0b	RSV	N/A	Reserved.
ISVLAN	9	0b	RW	UNDEFINED	**Is VLAN** If this bit is set, this tag is a VLAN tag. In this case, a tag with VLAN ID = 0 is treated as untagged (priority tagging) and the priority bits can be extracted to the Rx-Descriptor.
INNERUP	10	0b	RW	UNDEFINED	**Inner UP** If this bit is set, the UP remapping is done on this field. Should be set only on one tag. If this bit is set, the *ISVLAN* bit should also be set.
OUTERUP	11	0b	RW	UNDEFINED	**Outer UP** If this bit is set, this is the tag on which the inner-to-outer UP remapping is applied. Should be set only in one tag.
LONG	12	0b	RW	UNDEFINED	**Long** Do not support insert of entire header. In this case, the length can be higher than 8 bytes.
ISMPLS	13	0b	RW	UNDEFINED	**Is MPLS** Identifies the tags that contain a MPLS EtherType. This bit should be set for tags 6 (MPLS unicast) and 7 (MPLS multicast).
ISNSH	14	0b	RW	UNDEFINED	**Is NSH** Identifies the tags that contains a NSH EtherType.
RESERVED	15	0b	RSV	N/A	Reserved.
ETHERTYPE	31:16	0x0	RW	UNDEFINED	**EtherType** The EtherType identifying the L2 tag.

13.2.2.9 PF - FlexiPipe Registers

Registers used to configure the generic parts of the FlexiPipe. Registers specific to the analyzer, switch, ACL or filters are described in their respective sections.

13.2.2.9.1 Force Profile ID - GL_PSTEXT_FORCE_PID[n] (0x0020E000 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
STATIC_PID	15:0	0x1	RW	UNDEFINED	**Static Profile ID** Selected Profile ID if *STATIC_PID_EN* is set.
RESERVED	30:16	0x0	RSV	N/A	Reserved.
STATIC_PID_EN	31	1b	RW	UNDEFINED	**Static Profile ID Enable** Enable static Profile ID selection.

13.2.2.9.2 Profile Level Selector - GL_PSTEXT_PLVL_SEL[n] (0x0020E00C + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PLVL_SEL	0	1b	RW	UNDEFINED	**Profile Level Select** 0b = Profile selected by L1 profile (CDID). 1b = Profile selected by L2 profile.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.9.3 L1 Force CDID - GL_PSTEXT_FORCE_L1CDID[n] (0x0020E018 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
STATIC_CDID	3:0	0x0	RW	UNDEFINED	**Static CDID** Static CDID index to be used when static CDID is enabled.
RESERVED	30:4	0x0	RSV	N/A	Reserved.
STATIC_CDID_EN	31	0b	RW	UNDEFINED	**Static CDID Enable** Enable static CDID selection.

13.2.2.9.4 L1 P2P Table Configuration Address - GL_PSTEXT_P2P_L1ADDR[n] (0x0020E024 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LINE_IDX	0	0b	RW	UNDEFINED	**Line Index** XLT0 SRAM output to XLT0_CDID translation table entry address.
RESERVED	30:1	0x0	RSV	N/A	Reserved.
AUTO_INC	31	0b	RW	UNDEFINED	**Auto-Increment** Enable auto-increment. When this bit is set, the hardware automatically increments the *LINE_IDX* value on every P2P_L1DATA write/read.

13.2.2.9.5 L1 P2P Table Configuration Data - GL_PSTEXT_P2P_L1DATA[n] (0x0020E030 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DATA	31:0	0x0	RW	UNDEFINED	**Data** XLT0 SRAM output to XLT0_CDID translation table entry data

13.2.2.9.6 XLT0 Table Configuration Address - GL_PSTEXT_XLT0_L1ADDR[n] (0x0020E03C + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LINE_IDX	7:0	0x0	RW	UNDEFINED	**Line Index** XLT0 memory address.
RESERVED	30:8	0x0	RSV	N/A	Reserved.
AUTO_INC	31	0b	RW	UNDEFINED	**Auto-Increment** Enable auto-increment. When this bit is set, the hardware automatically increments the *LINE_IDX* on every XLT0_L1DATA write/read.

13.2.2.9.7 XLT0 Table Configuration Data - GL_PSTEXT_XLT0_L1DATA[n] (0x0020E048 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DATA	31:0	0x0	RW	UNDEFINED	**Data** XLT0 entry data.

13.2.2.9.8 L1 Bidirectional CTL - GL_PSTEXT_CDMD_L1SEL[n] (0x0020E054 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RX_SEL	4:0	0x0	RW	UNDEFINED	**Rx Select** Selects which MDID is used to select md0 input to XLT0 for Rx traffic.
RESERVED	7:5	000b	RSV	N/A	Reserved.
TX_SEL	12:8	0x0	RW	UNDEFINED	**Tx Select** Selects which MDID is used to select md0 input to XLT0 for Tx traffic.
RESERVED	15:13	000b	RSV	N/A	Reserved.
AUX0_SEL	20:16	0x0	RW	UNDEFINED	**Aux 0 Select** Selects which MDID is used to select md0 input to XLT0 for Rx traffic.
RESERVED	23:21	000b	RSV	N/A	Reserved.
AUX1_SEL	28:24	0x0	RW	UNDEFINED	**Aux 1 Select** Selects which MDID is used to select md0 input to XLT0 for Rx traffic
RESERVED	29	0b	RSV	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
BIDIR_ENA	31:30	00b	RW	UNDEFINED	**Bidirectional Enable** 00b = Direction not part of XLT0 input. 01b = LSB bit of direction (GL_PSTEXT_FLGS_L1TBL output) is part of XLT0 input. 10b = Two bits of direction (GL_PSTEXT_FLGS_L1TBL output) are part of XLT0 input. 11b = Reserved.

13.2.2.9.9 L1 Flag Select Table - GL_PSTEXT_FLGS_L1TBL[n] (0x0020E060 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LSB	15:0	0x0	RW	UNDEFINED	**LSB** A 4b->1b (16 entry) table maps the Flags values into a direction indication (LSB).
MSB	31:16	0x0	RW	UNDEFINED	**MSB** A 4b->1b (16 entry) table maps the Flags values into a direction indication (MSB).

13.2.2.9.10 L1 Flag Select Control (0-1) - GL_PSTEXT_FLGS_L1SEL0_1[n] (0x0020E06C + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FLS0	8:0	0x0	RW	UNDEFINED	**Flag Select 0** Index of first flag used to define the direction from list of flags described in the "MDID Namespace" section.
RESERVED	15:9	0x0	RSV	N/A	Reserved.
FLS1	24:16	0x0	RW	UNDEFINED	**Flag Select 1** Index of second flag used to define the direction from list of flags described in the "MDID Namespace" section
RESERVED	31:25	0x0	RSV	N/A	Reserved.

13.2.2.9.11 L1 Flag Select Control (2-3) - GL_PSTEXT_FLGS_L1SEL2_3[n] (0x0020E078 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FLS2	8:0	0x0	RW	UNDEFINED	**Flag Select 2** Index of third flag used to define the direction from list of flags described in the "MDID Namespace" section.
RESERVED	15:9	0x0	RSV	N/A	Reserved.
FLS3	24:16	0x0	RW	UNDEFINED	**Flag Select 3** Index of fourth flag used to define the direction from list of flags described in the "MDID Namespace" section
RESERVED	31:25	0x0	RSV	N/A	Reserved.

13.2.2.9.12 L2 Configuration Table Address - GL_PSTEXT_CTLTBL_L2ADDR[n] (0x0020E084 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LINE_OFF	2:0	000b	RW	UNDEFINED	**Line Offset** Data selection for CTLTBL_L2DATA (see Section 13.2.2.9.13). Valid values are 0...4.
RESERVED	7:3	0x0	RSV	N/A	Reserved.
LINE_IDX	10:8	000b	RW	UNDEFINED	**Line Index** CDID selection for MDID digest builder configuration.
RESERVED	30:11	0x0	RSV	N/A	Reserved.
AUTO_INC	31	0b	RW	UNDEFINED	**Auto-Increment** Auto-increment enable. When this bit is set, the hardware auto-increments the *LINE_OFF* for every write/read of CTLTBL_L2DATA, while incrementing *LINE_IDX* each time *LINE_OFF* reaches 0x4. *LINE_OFF* wraps around on 0x4 value.

13.2.2.9.13 L2 Configuration Table Data - GL_PSTEXT_CTLTBL_L2DATA[n] (0x0020E090 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DATA	31:0	0x0	RW	UNDEFINED	**Data** CTLTBL data according to CTLTBL_L2ADDR.*LINE_OFF*. For *LINE_OFF* = 0x0 [2:0] = xlt1_adsel [5:3] = xlt2_adsel [14:6] = flg0_sel [23:15] = flg1_sel [31:24] = flg2_sel[7:0] For *LINE_OFF* = 0x1 [0:0] = flg2_sel[5] [9:1] = flg3_sel [10:18] = flg4_sel [27:19] = flg5_sel [31:28] = flg6_sel[3:0] For *LINE_OFF* = 0x2 [4:0] = flg6_sel[8:4] [13:5] = flg7_sel [22:14] = flg8_sel [31:23] = flg9_sel For *LINE_OFF* = 0x3 [8:0] = flg10_sel [17:9] = flg11_sel [26:18] = flg12_sel [31:27] = flg13_sel[4:0] For *LINE_OFF* = 0x4 [3:0] = flg13_sel[5:2] [12:4] = flg14_sel [21:13] = Reserved [26:22] = xlt2_mdsel [31:27] = xlt1_mdsel

13.2.2.9.14 XLT1, XLT2 Partition Mode - GL_PSTEXT_L2PRTMOD[n] (0x0020E09C + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
XLT1	1:0	00b	RW	UNDEFINED	**XLT1** Selects the size of the partitions of XLT1: 00b = Single 8K entries partition. 01b = Two 4K entries partitions (selection by xlt1_adsel[0]). 10b = Four 2K entries partitions (selection by xlt1_adsel[1:0]. 11b = Eight 1K entries partitions (selection by xlt1_adsel[2:0].
RESERVED	7:2	0x0	RSV	N/A	Reserved.
XLT2	9:8	00b	RW	UNDEFINED	**XLT2** Selects the size of the partitions of XLT2: 00b = Single 1K entries partition. 01b = Two 512 entries partitions (selection by xlt2_adsel[0]). 10b = Four 256 entries partitions (selection by xlt2_adsel[1:0]. 11b = Eight 128 entries partitions (selection by xlt2_adsel[2:0].
RESERVED	31:10	0x0	RSV	N/A	Reserved.

13.2.2.9.15 XLT1 Table Configuration Address - GL_PSTEXT_XLT1_L2ADDR[n] (0x0020E0C0 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LINE_IDX	10:0	0x0	RW	UNDEFINED	**Line Index** XLT1 memory address.
RESERVED	30:11	0x0	RSV	N/A	Reserved.
AUTO_INC	31	0b	RW	UNDEFINED	**Auto-Increment** Enable auto-increment. When this bit is set, the hardware automatically increments the *LINE_IDX* on every XLT1_L2DATA write/read.

13.2.2.9.16 XLT1 Table Configuration Data - GL_PSTEXT_XLT1_L2DATA[n] (0x0020E0CC + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DATA	31:0	0x0	RW	UNDEFINED	**Data** XLT1 memory entry data. Four consequent entries are read/written at once: [7:0] = Offset 0 [15: 8] = Offset 1 [23:16] = Offset 2 [31:24] = Offset 3

13.2.2.9.17 XLT2 Table Configuration Address - GL_PSTEXT_XLT2_L2ADDR[n] (0x0020E0D8 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LINE_IDX	8:0	0x0	RW	UNDEFINED	**Line Index** XLT2 memory entry address.
RESERVED	30:9	0x0	RSV	N/A	Reserved.
AUTO_INC	31	0b	RW	UNDEFINED	**Auto-Increment** Enable auto-increment. When this bit is set, the hardware automatically increments the *LINE_IDX* on every XLT2_L2DATA write/read.

13.2.2.9.18 XLT2 Table Configuration Data - GL_PSTEXT_XLT2_L2DATA[n] (0x0020E0E4 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DATA	31:0	0x0	RW	UNDEFINED	**Data** XLT2 memory entry data. Two consequent entries are read/written at once: [15:0] = Offset 0 [31:16] = Offset 1

13.2.2.9.19 Profile ID Gen Key Type - GL_PSTEXT_PID_L2GKTYPE[n] (0x0020E0F0 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PID_GKTYPE	1:0	00b	RW	UNDEFINED	**Profile ID Gen Key Type** 00b = CD bitmap not part of key. 01b = 2 bits of CD bitmap are part of key. 10b = 4 bits of CD bitmap are part of key. 11b = 8 bits of CD bitmap are part of key.
RESERVED	31:2	0x0	RSV	N/A	Reserved.

13.2.2.9.20 Profile Key Mask (LSB) - GL_PSTEXT_L2_PMASK0[n] (0x0020E0FC + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BITMASK	31:0	0x0	RW	UNDEFINED	**Bitmask** MDID digest mask (32 LSBs).

13.2.2.9.21 Profile Key Mask (MSB) - GL_PSTEXT_L2_PMASK1[n] (0x0020E108 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BITMASK	15:0	0x0	RW	UNDEFINED	**Bitmask** MDID digest mask (16 MSBs).
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.9.22 TCAM Configuration (Address) - GL_PSTEXT_TCAM_L2ADDR[n] (0x0020E114 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LINE_IDX	9:0	0x0	RW	UNDEFINED	**Line Index** TCAM entry index.
RESERVED	30:10	0x0	RSV	N/A	Reserved.
AUTO_INC	31	0b	RW	UNDEFINED	**Auto-Increment** Enable auto-increment. When this bit is set, the hardware automatically increments the *LINE_IDX* on every TCAM_L2DATAMSB write/read.

13.2.2.9.23 TCAM Configuration LSB (Data) - GL_PSTEXT_TCAM_L2DATALSB[n] (0x0020E120 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DATALSB	31:0	0x0	RW	UNDEFINED	**Data LSB** TCAM entry/inverted mask data (LSB).

13.2.2.9.24 TCAM Configuration MSB (Data+Mask) - GL_PSTEXT_TCAM_L2DATAMSB[n] (0x0020E12C + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DATAMSB	7:0	0x0	RW	UNDEFINED	**Data MSB** TCAM entry/inverted mask data (MSB).
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.9.25 L2 Default Profile - GL_PSTEXT_DFLT_L2PRFL[n] (0x0020E138 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DFLT_PRFL	15:0	0x1	RW	UNDEFINED	**Default Profile** Profile IDX chosen when “no hit” in the TCAM.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.9.26 K2N Table Configuration Address - GL_PSTEXT_K2N_L2ADDR[n] (0x0020E144 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LINE_IDX	6:0	0x0	RW	UNDEFINED	**Line Index** TCAM entry to Profile ID mapping entry index (each index serves four TCAM entries).
RESERVED	30:7	0x0	RSV	N/A	Reserved.
AUTO_INC	31	0b	RW	UNDEFINED	**Auto-Increment** Enable auto-increment. When this bit is set, the hardware automatically increments the *LINE_IDX* on every K2N_L2DATA write/read.

13.2.2.9.27 K2N Table Configuration Data - GL_PSTEXT_K2N_L2DATA[n] (0x0020E150 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DATA0	7:0	0x0	RW	UNDEFINED	**Data 0** TCAM entry #(4*K2N_L2ADDR.*LINE_IDX*) destination Profile ID.
DATA1	15:8	0x0	RW	UNDEFINED	**Data 1** TCAM entry #(4*K2N_L2ADDR.*LINE_IDX*+1) destination Profile ID.
DATA2	23:16	0x0	RW	UNDEFINED	**Data 2** TCAM entry #(4*K2N_L2ADDR.*LINE_IDX*+2) destination Profile ID.
DATA3	31:24	0x0	RW	UNDEFINED	**Data 3** TCAM entry #(4*K2N_L2ADDR.*LINE_IDX*+3) destination Profile ID.

13.2.2.9.28 K2N Table Configuration Address - GL_PSTEXT_N2N_L2ADDR[n] (0x0020E15C + 0x4*n, n=0...2; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LINE_IDX	5:0	0x0	RW	UNDEFINED	**Line Index** N2N table mapping entry index (each index serves four entries).
RESERVED	30:6	0x0	RSV	N/A	Reserved.
AUTO_INC	31	0b	RW	UNDEFINED	**Auto-Increment** Enable auto-increment. When this bit is set, the hardware automatically increments the *LINE_IDX* on every K2N_L2DATA write/read.

13.2.2.9.29 K2N Table Configuration Data - GL_PSTEXT_N2N_L2DATA[n] (0x0020E168 + 0x4*n, n=0...2; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DATA0	7:0	0x0	RW	UNDEFINED	**Data 0** K2N entry #(4*N2N_L2ADDR.*LINE_IDX*) destination Profile ID.
DATA1	15:8	0x0	RW	UNDEFINED	**Data 1** K2N entry #(4*N2N_L2ADDR.*LINE_IDX*+1) destination Profile ID.
DATA2	23:16	0x0	RW	UNDEFINED	**Data 2** K2N entry #(4*N2N_L2ADDR.*LINE_IDX*+2) destination Profile ID.
DATA3	31:24	0x0	RW	UNDEFINED	**Data 3** K2N entry #(4*N2N_L2ADDR.*LINE_IDX*+3) destination Profile ID.

13.2.2.9.30 Profile Memory Configuration Data - GL_PSTEXT_PRFLM_DATA_0[n] (0x0020E174 + 0x4*n, n=0...63; RO)

This register array is used by the PE to program QH filter entries.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PROT	7:0	0x0	RW	UNDEFINED	**Protocol ID** Protocol ID in this FV location.
RESERVED	15:8	0x0	RSV	N/A	Reserved.
OFF	24:16	0x0	RW	UNDEFINED	**Offset** Offset within protocol header.
RESERVED	31:25	0x0	RSV	N/A	Reserved.

13.2.2.9.31 Profile Memory Configuration Data - GL_PSTEXT_PRFLM_DATA_1[n] (0x0020E274 + 0x4*n, n=0...63; RO)

This register array is used by the PE to program QH filter entries.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PROT	7:0	0x0	RW	UNDEFINED	**Protocol ID** Protocol ID in this FV location.
RESERVED	15:8	0x0	RSV	N/A	Reserved.
OFF	24:16	0x0	RW	UNDEFINED	**Offset** Offset within protocol header.
RESERVED	31:25	0x0	RSV	N/A	Reserved.

13.2.2.9.32 Profile Memory Configuration Data - GL_PSTEXT_PRFLM_DATA_2[n] (0x0020E374 + 0x4*n, n=0...63; RO)

This register array is used by the PE to program QH filter entries.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PROT	7:0	0x0	RW	UNDEFINED	**Protocol ID** Protocol ID in this FV location.
RESERVED	15:8	0x0	RSV	N/A	Reserved.
OFF	24:16	0x0	RW	UNDEFINED	**Offset** Offset within protocol header.
RESERVED	31:25	0x0	RSV	N/A	Reserved.

13.2.2.9.33 Profile Memory Configuration Control - GL_PSTEXT_PRFLM_CTRL[n] (0x0020E474 + 0x4*n, n=0...2; RO)

This register is used by the PE to program QH filter entries.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRFL_IDX	7:0	0x0	RW	UNDEFINED	**Profile Index** Programmed/read profile index.
RESERVED	29:8	0x0	RSV	N/A	Reserved.
RD_REQ	30	0b	RW	UNDEFINED	**Read Req** Setting the *RD_REQ* flag by the programming entity, the Profile "recipe" is written to GL_PSTEXT_PRFLM_DATA_[n] array, n=0..2. The *RD_REQ* flag is auto-cleared when the programming is received from the Profile "Recipe" Memory.
WR_REQ	31	0b	RW	UNDEFINED	**Write Req** Setting the *WR_REQ* flag by the programming entity, the Profile "recipe" is programmed by data previously written to GL_PSTEXT_PRFLM_DATA_[n] array, n=0..2. The *WR_REQ* flag is auto-cleared when the programming is sent to the Profile "Recipe" Memory.

13.2.2.9.34 PG L2 Flag15 Bitmask (LSB) - GL_PSTEXT_FL15_BMPLSB[n] (0x0020E480 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BMPLSB	31:0	0x0	RW	UNDEFINED	**Bitmap LSB** 32 LSBs of bitmask used for "flag 15" calculation.

13.2.2.9.35 PG L2 Flag15 Bitmask (MSB) - GL_PSTEXT_FL15_BMPMSB[n] (0x0020E48C + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BMPMSB	31:0	0x0	RW	UNDEFINED	**Bitmap MSB** 32 MSBs of bitmask used for "flag 15" calculation.

13.2.2.9.36 Profile Key Mask (LSB) - GL_PSTEXT_L2_TMASK0[n] (0x0020E498 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BITMASK	31:0	0x0	RW	UNDEFINED	**Bitmask** TCAM input mask (32 LSBs).

13.2.2.9.37 Profile Key Mask (MSB) - GL_PSTEXT_L2_TMASK1[n] (0x0020E4A4 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BITMASK	7:0	0x0	RW	UNDEFINED	**Bitmask** TCAM input mask (8 MSBs).
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.9.38 Force Profile ID - GL_PREEXT_FORCE_PID[n] (0x0020F000 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.1.

13.2.2.9.39 Profile Level Selector - GL_PREEXT_PLVL_SEL[n] (0x0020F00C + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.2.

13.2.2.9.40 L1 Force CDID - GL_PREEXT_FORCE_L1CDID[n] (0x0020F018 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.3.

13.2.2.9.41 L1 P2P Table Configuration Address - GL_PREEXT_P2P_L1ADDR[n] (0x0020F024 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.4.

13.2.2.9.42 L1 P2P Table Configuration Data - GL_PREEXT_P2P_L1DATA[n] (0x0020F030 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.5.

13.2.2.9.43 XLT0 Table Configuration Address - GL_PREEXT_XLT0_L1ADDR[n] (0x0020F03C + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.6.

13.2.2.9.44 XLT0 Table Configuration Data - GL_PREEXT_XLT0_L1DATA[n] (0x0020F048 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.7.

13.2.2.9.45 L1 Bidirectional CTL - GL_PREEXT_CDMD_L1SEL[n] (0x0020F054 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.8.

13.2.2.9.46 L1 Flag Select Table - GL_PREEXT_FLGS_L1TBL[n] (0x0020F060 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.9.

13.2.2.9.47 L1 Flag Select Control (0-1) - GL_PREEXT_FLGS_L1SEL0_1[n] (0x0020F06C + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.10.

13.2.2.9.48 L1 Flag Select Control (2-3) - GL_PREEXT_FLGS_L1SEL2_3[n] (0x0020F078 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.11.

13.2.2.9.49 L2 Configuration Table Address - GL_PREEXT_CTLTBL_L2ADDR[n] (0x0020F084 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.12.

13.2.2.9.50 L2 Configuration Table Data - GL_PREEXT_CTLTBL_L2DATA[n] (0x0020F090 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.13.

13.2.2.9.51 XLT1, XLT2 Partition Mode - GL_PREEXT_L2PRTMOD[n] (0x0020F09C + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.14.

13.2.2.9.52 PG L2 CDID Bitmap (LSB) - GL_PREEXT_L2BMP0_3[n] (0x0020F0A8 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BMP0	7:0	0x1	RW	UNDEFINED	**Bitmap 0** One-hot encoding for CDID 0.
BMP1	15:8	0x2	RW	UNDEFINED	**Bitmap 1** One-hot encoding for CDID 1.
BMP2	23:16	0x4	RW	UNDEFINED	**Bitmap 2** One-hot encoding for CDID 2.
BMP3	31:24	0x8	RW	UNDEFINED	**Bitmap 3** One-hot encoding for CDID 3.

13.2.2.9.53 PG L2 CDID Bitmap (MSB) - GL_PREEXT_L2BMP4_7[n] (0x0020F0B4 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BMP4	7:0	0x10	RW	UNDEFINED	**Bitmap 4** One-hot encoding for CDID 4.
BMP5	15:8	0x20	RW	UNDEFINED	**Bitmap 5** One-hot encoding for CDID 5.
BMP6	23:16	0x40	RW	UNDEFINED	**Bitmap 6** One-hot encoding for CDID 6.
BMP7	31:24	0x80	RW	UNDEFINED	**Bitmap 7** One-hot encoding for CDID 7.

13.2.2.9.54 XLT1 Table Configuration Address - GL_PREEXT_XLT1_L2ADDR[n] (0x0020F0C0 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.15.

13.2.2.9.55 XLT1 Table Configuration Data - GL_PREEXT_XLT1_L2DATA[n] (0x0020F0CC + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.16.

13.2.2.9.56 XLT2 Table Configuration Address - GL_PREEXT_XLT2_L2ADDR[n] (0x0020F0D8 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.17.

13.2.2.9.57 XLT2 Table Configuration Data - GL_PREEXT_XLT2_L2DATA[n] (0x0020F0E4 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.18.

13.2.2.9.58 Profile ID Gen Key Type - GL_PREEXT_PID_L2GKTYPE[n] (0x0020F0F0 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.19.

13.2.2.9.59 Profile Key Mask (LSB) - GL_PREEXT_L2_PMASK0[n] (0x0020F0FC + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.20.

13.2.2.9.60 Profile Key Mask (MSB) - GL_PREEXT_L2_PMASK1[n] (0x0020F108 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.21.

13.2.2.9.61 TCAM Configuration (Address) - GL_PREEXT_TCAM_L2ADDR[n] (0x0020F114 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.22.

13.2.2.9.62 TCAM Configuration LSB (Data) - GL_PREEXT_TCAM_L2DATALSB[n] (0x0020F120 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.23.

13.2.2.9.63 TCAM Configuration MSB (Data+Mask) - GL_PREEXT_TCAM_L2DATAMSB[n] (0x0020F12C + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.24.

13.2.2.9.64 L2 Default Profile - GL_PREEXT_DFLT_L2PRFL[n] (0x0020F138 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.25.

13.2.2.9.65 K2N Table Configuration Address - GL_PREEXT_K2N_L2ADDR[n] (0x0020F144 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.26.

13.2.2.9.66 K2N Table Configuration Data - GL_PREEXT_K2N_L2DATA[n] (0x0020F150 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.27.

13.2.2.9.67 K2N Table Configuration Address - GL_PREEXT_N2N_L2ADDR[n] (0x0020F15C + 0x4*n, n=0...2; RW)

Field definitions are the same as those defined in Section 13.2.2.9.28.

13.2.2.9.68 K2N Table Configuration Data - GL_PREEXT_N2N_L2DATA[n] (0x0020F168 + 0x4*n, n=0...2; RW)

Field definitions are the same as those defined in Section 13.2.2.9.29.

13.2.2.9.69 Profile Key Mask (LSB) - GL_PREEXT_L2_TMASK0[n] (0x0020F498 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.36.

13.2.2.9.70 Profile Key Mask (MSB) - GL_PREEXT_L2_TMASK1[n] (0x0020F4A4 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.37.

13.2.2.9.71 Force Profile ID - GL_ACLEXT_FORCE_PID[n] (0x00210000 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.1.

13.2.2.9.72 Profile Level Selector - GL_ACLEXT_PLVL_SEL[n] (0x0021000C + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.2.

13.2.2.9.73 L1 Force CDID - GL_ACLEXT_FORCE_L1CDID[n] (0x00210018 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.3.

13.2.2.9.74 L1 P2P Table Configuration Address - GL_ACLEXT_P2P_L1ADDR[n] (0x00210024 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.4.

13.2.2.9.75 L1 P2P Table Configuration Data - GL_ACLEXT_P2P_L1DATA[n] (0x00210030 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.5.

13.2.2.9.76 XLT0 Table Configuration Address - GL_ACLEXT_XLT0_L1ADDR[n] (0x0021003C + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.6.

13.2.2.9.77 XLT0 Table Configuration Data - GL_ACLEXT_XLT0_L1DATA[n] (0x00210048 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.7.

13.2.2.9.78 L1 Bidirectional CTL - GL_ACLEXT_CDMD_L1SEL[n] (0x00210054 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.8.

13.2.2.9.79 L1 Flag Select Table - GL_ACLEXT_FLGS_L1TBL[n] (0x00210060 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.9.

13.2.2.9.80 L1 Flag Select Control (0-1) - GL_ACLEXT_FLGS_L1SEL0_1[n] (0x0021006C + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.10.

13.2.2.9.81 L1 Flag Select Control (2-3) - GL_ACLEXT_FLGS_L1SEL2_3[n] (0x00210078 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.11.

13.2.2.9.82 L2 Configuration Table Address - GL_ACLEXT_CTLTBL_L2ADDR[n] (0x00210084 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.12.

13.2.2.9.83 L2 Configuration Table Data - GL_ACLEXT_CTLTBL_L2DATA[n] (0x00210090 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.13.

13.2.2.9.84 XLT1, XLT2 Partition Mode - GL_ACLEXT_L2PRTMOD[n] (0x0021009C + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.14.

13.2.2.9.85 PG L2 CDID Bitmap (LSB) - GL_ACLEXT_L2BMP0_3[n] (0x002100A8 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BMP0	7:0	0x1	RW	UNDEFINED	**Bitmap 0** One-hot encoding for CDID 0.
BMP1	15:8	0x2	RW	UNDEFINED	**Bitmap 1** One-hot encoding for CDID 1.
BMP2	23:16	0x4	RW	UNDEFINED	**Bitmap 2** One-hot encoding for CDID 2.
BMP3	31:24	0x8	RW	UNDEFINED	**Bitmap 3** One-hot encoding for CDID 3.

13.2.2.9.86 PG L2 CDID Bitmap (MSB) - GL_ACLEXT_L2BMP4_7[n] (0x002100B4 + 0x4*n, n=0...2; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BMP4	7:0	0x10	RW	UNDEFINED	**Bitmap 4** One-hot encoding for CDID 4.
BMP5	15:8	0x20	RW	UNDEFINED	**Bitmap 5** One-hot encoding for CDID 5.
BMP6	23:16	0x40	RW	UNDEFINED	**Bitmap 6** One-hot encoding for CDID 6.
BMP7	31:24	0x80	RW	UNDEFINED	**Bitmap 7** One-hot encoding for CDID 7.

13.2.2.9.87 XLT1 Table Configuration Address - GL_ACLEXT_XLT1_L2ADDR[n] (0x002100C0 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.15.

13.2.2.9.88 XLT1 Table Configuration Data - GL_ACLEXT_XLT1_L2DATA[n] (0x002100CC + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.16.

13.2.2.9.89 XLT2 Table Configuration Address - GL_ACLEXT_XLT2_L2ADDR[n] (0x002100D8 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.17.

13.2.2.9.90 XLT2 Table Configuration Data - GL_ACLEXT_XLT2_L2DATA[n] (0x002100E4 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.18.

13.2.2.9.91 Profile ID Gen Key Type - GL_ACLEXT_PID_L2GKTYPE[n] (0x002100F0 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.19.

13.2.2.9.92 Profile Key Mask (LSB) - GL_ACLEXT_L2_PMASK0[n] (0x002100FC + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.20.

13.2.2.9.93 Profile Key Mask (MSB) - GL_ACLEXT_L2_PMASK1[n] (0x00210108 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.21.

13.2.2.9.94 TCAM Configuration (Address) - GL_ACLEXT_TCAM_L2ADDR[n] (0x00210114 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.22.

13.2.2.9.95 TCAM Configuration LSB (Data) - GL_ACLEXT_TCAM_L2DATALSB[n] (0x00210120 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.23.

13.2.2.9.96 TCAM Configuration MSB (Data) - GL_ACLEXT_TCAM_L2DATAMSB[n] (0x0021012C + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.24.

13.2.2.9.97 L2 Default Profile - GL_ACLEXT_DFLT_L2PRFL[n] (0x00210138 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.25.

13.2.2.9.98 K2N Table Configuration Address - GL_ACLEXT_K2N_L2ADDR[n] (0x00210144 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.26.

13.2.2.9.99 K2N Table Configuration Data - GL_ACLEXT_K2N_L2DATA[n] (0x00210150 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.27.

13.2.2.9.100 K2N Table Configuration Address - GL_ACLEXT_N2N_L2ADDR[n] (0x0021015C + 0x4*n, n=0...2; RW)

Field definitions are the same as those defined in Section 13.2.2.9.28.

13.2.2.9.101 K2N Table Configuration Data - GL_ACLEXT_N2N_L2DATA[n] (0x00210168 + 0x4*n, n=0...2; RW)

Field definitions are the same as those defined in Section 13.2.2.9.29.

13.2.2.9.102 Profile Key Mask (LSB) - GL_ACLEXT_L2_TMASK0[n] (0x00210498 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.36.

13.2.2.9.103 Profile Key Mask (MSB) - GL_ACLEXT_L2_TMASK1[n] (0x002104A4 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.37.

13.2.2.9.104 L2 Default Profile - GL_ACLEXT_DFLT_L2PRFL_ACL[n] (0x00393800 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.2.9.25.

13.2.2.9.105 Tx Scheduling Correction Control - GLFLXP_TX_SCHED_CORRECT[n,m] (0x00458000 + 0x4*n + 0x100*m, n=0...63, m=0...31; RO)

Controls the reporting of the Tx scheduling corrections from the Rx path, per operator.

The correlation between each field and each CSR within the operator is:

We have 4 corrections per operator, each using 16 tupples (tupple 0 being the highest priority tupple) of protID & recipe.

So:

Correction 0 uses m=0..7, where tupple0/tupple1 use m=0, tupple2/tupple3 use m=1,... tupple14/tupple15 use m=7.

Correction 1 uses m=8..15, where tupple0/tupple1 use m=8, tupple2/tupple3 use m=9,... tupple14/tupple15 use m=15

Correction 2 uses m=16..23, where tupple0/tupple1 use m=16, tupple2/tupple3 use m=17,... tupple14/tupple15 use m=23

Correction 3 uses m=24..31, where tupple0/tupple1 use m=24, tupple2/tupple3 use m=25,... tupple14/tupple15 use m=31

Field	Bit(s)	Init.	Type	CFG Policy	Description
PROTD_ID_2N	7:0	0x0	RW	UNDEFINED	**Protocol ID 2n** ProtID 2n of offset "offset" of operator "operator" to look for in the packet's reported protIDs using prioritization mechanism of "look first for lowest n".
RECIPE_2N	12:8	0x0	RW	UNDEFINED	**Recipe 2n** Recipe 2n of offset "offset" of operator "operator" to be reported in case the matching protID was found ("won" in the priority list) in the packet.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	15:13	000b	RSV	N/A	Reserved.
PROTD_ID_2N_1	23:16	0x0	RW	UNDEFINED	**Protocol ID 2n+1** ProtID 2n+1 of offset "offset" of operator "operator" to look for in the packet's reported protIDs using prioritization mechanism of "look first for lowest n".
RECIPE_2N_1	28:24	0x0	RW	UNDEFINED	**Recipe 2n+1** Recipe 2n+1 of offset "offset" of operator "operator" to be reported in case the matching protID was found ("won" in the priority list) in the packet.
RESERVED	31:29	000b	RSV	N/A	Reserved.

13.2.2.9.106 ProtIDs for Creating RRX CMD Offsets - GLFLXP_RX_CMD_PROTIDS[n,m] (0x0045A000 + 0x4*n + 0x400*m, n=0...255, m=0...5; RO)

Determines the protID for the L3 protocols to match the ones configured in the Analyzer (for later parsing in the Rx-Pipe).

Field	Bit(s)	Init.	Type	CFG Policy	Description
PROTID_4N	7:0	0x0	RW	UNDEFINED	**Protocol ID 4n** ProtID #0 for creating offset N to RDPU (out of the 6 RDPU offsets).
PROTID_4N_1	15:8	0x0	RW	UNDEFINED	**Protocol ID 4n+1** ProtID #1 for creating offset N to RDPU (out of the 6 RDPU offsets).
PROTID_4N_2	23:16	0x0	RW	UNDEFINED	**Protocol ID 4n+2** ProtID #2 for creating offset N to RDPU (out of the 6 RDPU offsets).
PROTID_4N_3	31:24	0x0	RW	UNDEFINED	**Protocol ID 4n+3** ProtID #3 for creating offset N to RDPU (out of the 6 RDPU offsets).

13.2.2.9.107 PTYPE_10b to PTYPE_8b Translation - GLFLXP_PTYPE_TRANSLATION[n] (0x0045C000 + 0x4*n, n=0...255; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PTYPE_4N	7:0	0x0	RW	UNDEFINED	**PTYPE 4n** PTYPE_8b for PTYPE_10b N.
PTYPE_4N_1	15:8	0x0	RW	UNDEFINED	**PTYPE 4n+1** PTYPE_8b for PTYPE_10b N+1.
PTYPE_4N_2	23:16	0x0	RW	UNDEFINED	**PTYPE 4n+2** PTYPE_8b for PTYPE_10b N+2.
PTYPE_4N_3	31:24	0x0	RW	UNDEFINED	**PTYPE 4n+3** PTYPE_8b for PTYPE_10b N+3.

13.2.2.9.108 LX Prot & Index for Rx CMD - GLFLXP_RX_CMD_LX_PROT_IDX[n] (0x0045C400 + 0x4*n, n=0...255; RO)

Determines the LX prot and cloud/L4/payload offsets index per 8b PTPYE for RDPU.

Field	Bit(s)	Init.	Type	CFG Policy	Description
INNER_CLOUD_OFFSET_INDEX	2:0	000b	RW	UNDEFINED	**Inner Cloud Offset Index** Index of "L3_header_offset_*" that defines first inner header of the packet (After Outer IP/GRE/ Outer UDP). Value of 000b means there is no cloud tunneling header.
RESERVED	3	0b	RSV	N/A	Reserved.
L4_OFFSET_INDEX	6:4	000b	RW	UNDEFINED	**L4 Offset Index** Index of "L3_header_offset_*" that defines the L4 header in the packet. Value of 000b means there is no L4 header.
RESERVED	7	0b	RSV	N/A	Reserved.
PAYLOAD_OFFSET_INDEX	10:8	000b	RW	UNDEFINED	**Payload Offset Index** Index of "L3_header_offset_*" that defines the payload in the packet. Value of 000b means there is no L4 header.
RESERVED	11	0b	RSV	N/A	Reserved.
L3_PROTOCOL	13:12	00b	RW	UNDEFINED	**L3 Protocol** Indicates the first L3 protocol of the packet: 00b = IP4 01b = IPv6 10b = Non-valid value 11b = Other
L4_PROTOCOL	15:14	00b	RW	UNDEFINED	**L4 Protocol** Indicates the last L4 protocol of the packet. 00b = TCP 01b = UDP 10b = SCTP 11b = Other
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.9.109 RXDID FlexiWord 0 Control - GLFLXP_RXDID_FLX_WRD_0[n] (0x0045C800 + 0x4*n, n=0...63; RO)

Controls the reporting of FlexiWord 0 in the flexible Rx-Descriptors, per RXDID. Default value is meant to be aligned with legacy behavior (RSS/FDID).

Field	Bit(s)	Init.	Type	CFG Policy	Description
PROT_MDID	7:0	0x38	RW	UNDEFINED	**Protocol ID or MDID** This field can hold a protID value or a MDID value (for flexiword[n] of RXDID_IDX "RXDID_idx"). Its decoding is based on the *RXDID_OPCODE* field. ProtID: For reporting its offset or for extracting pkt bytes from its offset. MDID: For reporting the MDID's content. Default value is aimed to serve the selection of RSS (low word), to match legacy behavior.

Field	Bit(s)	Init.	Type	CFG Policy	Description
EXTRACTION_OFFSET	17:8	0x0	RW	UNDEFINED	**Extraction Offset** Relevant only if *PROT_MDID* field is a ProtID. Used as offset within protID to extract bytes from the packet. Its usage is based on *RXDID_OPCODE* field.
RESERVED	29:18	0x0	RSV	N/A	Reserved.
RXDID_OPCODE	31:30	01b	RW	UNDEFINED	**RXDID Opcode** Opcode to decide what to report in flexword[n] into the Rx-Descriptor: 00b = Fixed Value — Report the value from *PROT_MDID* field itself (for debug purposes). 01b = Metadata Offset — Report the selected MD word from the internal FlexiPipe buses. 10b = Extraction Offset — Report the value of bytes (protID's offset + byte offset) in the packet. 11b = Protocol Offset — Report the offset to the selected protocol.

13.2.2.9.110 RXDID FlexiWord 1 Control - GLFLXP_RXDID_FLX_WRD_1[n] (0x0045C900 + 0x4*n, n=0...63; RO)

Controls the reporting of FlexiWord 1 in the flexible Rx-Descriptors, per RXDID. Default value is meant to be aligned with legacy behavior (RSS/FDID).

Field	Bit(s)	Init.	Type	CFG Policy	Description
PROT_MDID	7:0	0x39	RW	UNDEFINED	**Protocol ID or MDID** This field can hold a protID value or a MDID value (for flexiword[n] of RXDID_IDX "RXDID_idx"). Its decoding is based on the *RXDID_OPCODE* field. ProtID: For reporting its offset or for extracting pkt bytes from its offset. MDID: For reporting the MDID's content. Default value is aimed to serve the selection of RSS (high word), to match legacy behavior.
EXTRACTION_OFFSET	17:8	0x0	RW	UNDEFINED	**Extraction Offset** Relevant only if *PROT_MDID* field is a ProtID. Used as offset within protID to extract bytes from the packet. Its usage is based on *RXDID_OPCODE* field.
RESERVED	29:18	0x0	RSV	N/A	Reserved.
RXDID_OPCODE	31:30	01b	RW	UNDEFINED	**RXDID Opcode** Opcode to decide what to report in flexword[n] into the Rx-Descriptor: 00b = Fixed Value — Report the value from *PROT_MDID* field itself (for debug purposes). 01b = Metadata Offset — Report the selected MD word from the internal FlexiPipe buses. 10b = Extraction Offset — Report the value of bytes (protID's offset + byte offset) in the packet. 11b = Protocol Offset — Report the offset to the selected protocol.

13.2.2.9.111 RXDID FlexiWord 2 Control - GLFLXP_RXDID_FLX_WRD_2[n] (0x0045CA00 + 0x4*n, n=0...63; RO)

Controls the reporting of FlexiWord 2 in the flexible Rx-Descriptors, per RXDID. Default value is meant to be aligned with legacy behavior (RSS/FDID).

Field	Bit(s)	Init.	Type	CFG Policy	Description
PROT_MDID	7:0	0x0	RW	UNDEFINED	**Protocol ID or MDID** This field can hold a protID value or a MDID value (for flexiword[n] of RXDID_IDX "RXDID_idx"). Its decoding is based on the *RXDID_OPCODE* field. ProtID: For reporting its offset or for extracting pkt bytes from its offset. MDID: For reporting the MDID's content. Default = 0 (genericMD_word0.
EXTRACTION_OFFSET	17:8	0x0	RW	UNDEFINED	**Extraction Offset** Relevant only if *PROT_MDID* field is a ProtID. Used as offset within protID to extract bytes from the packet. Its usage is based on *RXDID_OPCODE* field.
RESERVED	29:18	0x0	RSV	N/A	Reserved.
RXDID_OPCODE	31:30	01b	RW	UNDEFINED	**RXDID Opcode** Opcode to decide what to report in flexword[n] into the Rx-Descriptor: 00b = Fixed Value — Report the value from *PROT_MDID* field itself (for debug purposes). 01b = Metadata Offset — Report the selected MD word from the internal FlexiPipe buses. 10b = Extraction Offset — Report the value of bytes (protID's offset + byte offset) in the packet. 11b = Protocol Offset — Report the offset to the selected protocol.

13.2.2.9.112 RXDID FlexiWord 3 Control - GLFLXP_RXDID_FLX_WRD_3[n] (0x0045CB00 + 0x4*n, n=0...63; RO)

Controls the reporting of FlexiWord 3 in the flexible Rx-Descriptors, per RXDID. Default value is meant to be aligned with legacy behavior (RSS/FDID).

Field	Bit(s)	Init.	Type	CFG Policy	Description
PROT_MDID	7:0	0x1	RW	UNDEFINED	**Protocol ID or MDID** This field can hold a protID value or a MDID value (for flexiword[n] of RXDID_IDX "RXDID_idx"). Its decoding is based on the *RXDID_OPCODE* field. ProtID: For reporting its offset or for extracting pkt bytes from its offset. MDID: For reporting the MDID's content. Default = 1 (genericMD_word1).
EXTRACTION_OFFSET	17:8	0x0	RW	UNDEFINED	**Extraction Offset** Relevant only if *PROT_MDID* field is a ProtID. Used as offset within protID to extract bytes from the packet. Its usage is based on *RXDID_OPCODE* field.
RESERVED	29:18	0x0	RSV	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RXDID_OPCODE	31:30	01b	RW	UNDEFINED	**RXDID Opcode** Opcode to decide what to report in flexword[n] into the Rx-Descriptor: 00b = Fixed Value — Report the value from *PROT_MDID* field itself (for debug purposes). 01b = Metadata Offset — Report the selected MD word from the internal FlexiPipe buses. 10b = Extraction Offset — Report the value of bytes (protID's offset + byte offset) in the packet. 11b = Protocol Offset — Report the offset to the selected protocol.

13.2.2.9.113 RXDID FlexiWord 4 Control - GLFLXP_RXDID_FLX_WRD_4[n] (0x0045CC00 + 0x4*n, n=0...63; RO)

Controls the reporting of FlexiWord 4 in the flexible Rx-Descriptors, per RXDID. Default value is meant to be aligned with legacy behavior (RSS/FDID).

Field	Bit(s)	Init.	Type	CFG Policy	Description
PROT_MDID	7:0	0x5	RW	UNDEFINED	**Protocol ID or MDID** This field can hold a protID value or a MDID value (for flexiword[n] of RXDID_IDX "RXDID_idx"). Its decoding is based on the *RXDID_OPCODE* field. ProtID: For reporting its offset or for extracting pkt bytes from its offset. MDID: For reporting the MDID's content. Default value is aimed to serve the selection of FlowID (low word), to match legacy behavior (called "FDFID" in legacy).
EXTRACTION_OFFSET	17:8	0x0	RW	UNDEFINED	**Extraction Offset** Relevant only if *PROT_MDID* field is a ProtID. Used as offset within protID to extract bytes from the packet. Its usage is based on *RXDID_OPCODE* field.
RESERVED	29:18	0x0	RSV	N/A	Reserved.
RXDID_OPCODE	31:30	01b	RW	UNDEFINED	**RXDID Opcode** Opcode to decide what to report in flexword[n] into the Rx-Descriptor: 00b = Fixed Value — Report the value from *PROT_MDID* field itself (for debug purposes). 01b = Metadata Offset — Report the selected MD word from the internal FlexiPipe buses. 10b = Extraction Offset — Report the value of bytes (protID's offset + byte offset) in the packet. 11b = Protocol Offset — Report the offset to the selected protocol.

13.2.2.9.114 RXDID FlexiWord 5 Control - GLFLXP_RXDID_FLX_WRD_5[n] (0x0045CD00 + 0x4*n, n=0...63; RO)

Controls the reporting of FlexiWord 5 in the flexible Rx-Descriptors, per RXDID. Default value is meant to be aligned with legacy behavior (RSS/FDID).

Field	Bit(s)	Init.	Type	CFG Policy	Description
PROT_MDID	7:0	0x6	RW	UNDEFINED	**Protocol ID or MDID** This field can hold a protID value or a MDID value (for flexiword[n] of RXDID_IDX "RXDID_idx"). Its decoding is based on the *RXDID_OPCODE* field. ProtID: For reporting its offset or for extracting pkt bytes from its offset. MDID: For reporting the MDID's content. Default value is aimed to serve the selection of FlowID (high word), to match legacy behavior (called "FDFID" in legacy).
EXTRACTION_OFFSET	17:8	0x0	RW	UNDEFINED	**Extraction Offset** Relevant only if *PROT_MDID* field is a ProtID. Used as offset within protID to extract bytes from the packet. Its usage is based on *RXDID_OPCODE* field.
RESERVED	29:18	0x0	RSV	N/A	Reserved.
RXDID_OPCODE	31:30	01b	RW	UNDEFINED	**RXDID Opcode** Opcode to decide what to report in flexword[n] into the Rx-Descriptor: 00b = Fixed Value — Report the value from *PROT_MDID* field itself (for debug purposes). 01b = Metadata Offset — Report the selected MD word from the internal FlexiPipe buses. 10b = Extraction Offset — Report the value of bytes (protID's offset + byte offset) in the packet. 11b = Protocol Offset — Report the offset to the selected protocol.

13.2.2.9.115 RXDID FlexiFlags Control - GLFLXP_RXDID_FLAGS[n,m] (0x0045D000 + 0x4*n + 0x100*m, n=0...63, m=0...4; RO)

Controls the reporting of the FlexiFlags in the flexible Rx-Descriptors.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FLEXIFLAG_4N	5:0	0x0	RW	UNDEFINED	**FlexiFlag 4n** Determines the flag index to be reported in flexiflag[4*m] of RXDID_IDX "n" from the 64 available flags in the FlexiPipe.
RESERVED	7:6	00b	RSV	N/A	Reserved.
FLEXIFLAG_4N_1	13:8	0x0	RW	UNDEFINED	**FlexiFlag 4n+1** Same for 4m+1.
RESERVED	15:14	00b	RSV	N/A	Reserved.
FLEXIFLAG_4N_2	21:16	0x0	RW	UNDEFINED	**FlexiFlag 4n+2** Same for 4m+2.
RESERVED	23:22	00b	RSV	N/A	Reserved.
FLEXIFLAG_4N_3	29:24	0x0	RW	UNDEFINED	**FlexiFlag 4n+3** Same for 4m+3.
RESERVED	31:30	00b	RSV	N/A	Reserved.

13.2.2.9.116 RXDID Flags1 Override Control - GLFLXP_RXDID_FLAGS1_OVERRIDE[n] (0x0045D600 + 0x4*n, n=0...63; RO)

Controls the overriding of the FlexiFlags1 in the flexible Rx-Descriptors with RDPU related information.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FLEXIFLAGS1_OVERIDE	3:0	0x0	RW	UNDEFINED	**FlexiFlags 1 Override** If Bit[0] is set, EXT_UDP_0 indication from RDPU replaces the value of flexiflags1[0]. If Bit[1] is set, INT_UDP_0 indication from RDPU replaces the value of flexiflags1[1]. If Bit[2] is set, RECIPE_ERROR indication from RDPU replaces the value of flexiflags1[2]. If Bit[3] is set: OVERSIZE indication from RDPU replaces the value of flexiflags1[3].
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.9.117 Queue Context Flex Extension - QRXFLXP_CNTXT[QRX] (0x00480000 + 0x4*QRX, QRX=0...2047; RW)

Extension to the legacy queue context, supporting RXDID and TimeSync information.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RXDID_IDX	5:0	0x0	RW	UNDEFINED	**RXDID Index** Default RXDID_idx for this queue.
RESERVED	7:6	00b	RSV	N/A	Reserved.
RXDID_PRIO	10:8	001b	RW	UNDEFINED	**RXDID Priority** Priority of this RXDID_idx to be compared against previous priority from the pipe. If equal, this one wins.
TS	11	0b	RW	UNDEFINED	**Timestamp** Timestamp reporting per Rx packet enabled for this queue.
RESERVED	31:12	0x0	RSV	N/A	Reserved.

13.2.2.10 PF - Parser Registers

13.2.2.10.1 PRS Balancer Config - GL_PRS_RX_SIZE_CTRL (0x00200004; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MIN_SIZE	9:0	0x3C	RW	UNDEFINED	**Minimum Size** Minimum frame size (bytes).
RESERVED	14:10	0x0	RSV	N/A	Reserved.
MIN_SIZE_EN	15	0b	RW	UNDEFINED	**Minimum Size Enable** Enable undersize (<49 bytes) frames padding.
MAX_SIZE	25:16	0x1F8	RW	UNDEFINED	**Maximum Size** Maximum frame size (bytes).
RESERVED	30:26	0x0	RSV	N/A	Reserved.
MAX_SIZE_EN	31	1b	RW	UNDEFINED	**Maximum Size Enable** 0b = Read entire packet. 1b = Enable read of header only. Must be set to 1b.

13.2.2.10.2 Rx-Query Pipe-Status Init for Word 0-6 - GL_PRS_RX_PIPE_INIT0[n] (0x0020000C + 0x4*n, n=0...6; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GPCSR_INIT	15:0	0xFFFF	RW	UNDEFINED	**GPCSR Initial** Initial value of this word in the pipe_status.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.10.3 Rx-Query Pipe-Status Init for Word 7 - GL_PRS_RX_PIPE_INIT1 (0x00200028; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GPCSR_INIT	15:0	0x0040	RW	UNDEFINED	**GPCSR Initial** Initial value of this word in the pipe_status.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.10.4 Rx-Query Pipe-Status Init for Word 8 - GL_PRS_RX_PIPE_INIT2 (0x0020002C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GPCSR_INIT	15:0	0x2492	RW	UNDEFINED	**GPCSR Initial** Initial value of this word in the pipe_status.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.10.5 XLR Marker Trigger - GL_XLR_MARKER_TRIG_RCU_PRS (0x002001C0; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VM_VF_NUM	9:0	0x0	RW	UNDEFINED	**VF/VM Number** The number of the VF/VM to be reset (invalid if *VM_VF_TYPE* = PF).
VM_VF_TYPE	11:10	00b	RW	UNDEFINED	**VF/VM Type** 00b = Reset for VF 01b = Reset for VM. 10b = Reset for PF only. 11b = Reserved.
PF_NUM	14:12	000b	RW	UNDEFINED	**PF Number** PF number of the reset operation (should be valid in VM/VF reset type as well).
RESERVED	15	0b	RSV	N/A	Reserved.
PORT_NUM	18:16	000b	RW	UNDEFINED	**Port Number** The port to which the function belongs.
RESERVED	31:19	0x0	RSV	N/A	Reserved.

13.2.2.10.6 QH Removal Marker Trigger - GL_QH_MARKER_TRIG_RCU_PRS[n] (0x002001C4 + 0x4*n, n=0...3; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
QPID	17:0	0x0	RW	UNDEFINED	**Queue Pair ID** RDMA queue pair ID.
PE_TAG	25:18	0x0	RW	UNDEFINED	**PE Tag** Internal PE tag.
PORT_NUM	28:26	000b	RW	UNDEFINED	**Port Number** The port to which the function belongs.
RESERVED	30:29	00b	RSV	N/A	Reserved.
SET_RST	31	0b	SC	UNDEFINED	**SET_RST** 0b = Event is cleared. 1b = Writing operation to this array (0..3) bit triggers marker injection to TCs under *PORT_NUM* port.

13.2.2.10.7 COTF Marker Trigger - GL_COTF_MARKER_TRIG_RCU_PRS[n] (0x002001D4 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SET_RST	0	0b	SC	UNDEFINED	**SET_RST** Writing operation to this array (0..7) triggers marker injection for all TCs. Cleared when marker is sent.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.10.8 XLR Debug Markers Status - GL_XLR_MARKER_STATUS[n] (0x002001F4 + 0x4*n, n=0...1; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MRKR_BUSY	31:0	0x0	RO	N/A	**Marker Busy** Each respective bit represents a pending marker injection, by order of triggering. When 1, indicates marker has not been fully sent yet. 0 means sent. From 2 CSRs - 40 LSb are valid. This CSR should be read before initiating a new XLR marker, to make sure that at least one bit here signals "not-busy".

13.2.2.10.9 QH Debug Markers Status - GL_QH_MARKER_STATUS (0x002001FC; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MRKR_BUSY	3:0	0x0	RO	N/A	**Marker Busy** Each respective bit represents a pending marker injection, by order of triggering. When 1, indicates marker has not been fully sent yet. 0 means sent. This CSR should be read before initiating a new PE marker, to make sure that the index that will be used is indeed "not-busy".
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.10.10 COTF Debug Markers Status - GL_COTF_MARKER_STATUS (0x00200200; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MRKR_BUSY	7:0	0x0	RO	N/A	**Marker Busy** Each respective bit represents a pending marker injection, by order of triggering. When 1, indicates marker has not been fully sent yet. 0 means sent. This CSR should be read before initiating a new COTF marker, to make sure that the index that will be used is indeed "not-busy".
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.10.11 PRS Markers Error Indication (Marker FIFO Full) - GL_PRS_MARKER_ERROR (0x00200204; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
XLR_CFG_ERR	0	0b	RCW	UNDEFINED	**XLR Config Error** Sticky bit. Clears on read.
QH_CFG_ERR	1	0b	RCW	UNDEFINED	**QH Config Error** Sticky bit. Clears on read.
COTF_CFG_ERR	2	0b	RCW	UNDEFINED	**COTF Config Error** Sticky bit. Clears on read.
RESERVED	31:3	0x0	RSV	N/A	Reserved.

13.2.2.10.12 PRS Marker FIFO Read Access - GL_XLR_MARKER_LOG_RCU_PRS[n] (0x00200208 + 0x4*n, n=0...63; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
XLR_TRIG	31:0	0x0	RO	N/A	**XLR Trigger** Each array index relevant to the configured XLR trigger index. Each CSR contains exact write data as logged in CSR write access to *XLR_MARKER_TRIG*.

13.2.2.10.13 Rx ANA CSR Access Control - GL_RPRS_ANA_CSR_CTRL (0x00200708; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SELECT_EN	0	0b	RW	UNDEFINED	**Select Enable** Enable single selected access to ANA0/1. 0b = CSR access to ANA register is broadcasted to both analyzers. Read data is driven only by ANA0. 1b = CSR access is pointed to a single analyzer according to *SELECTED_ANA* bit.
SELECTED_ANA	1	0b	RW	UNDEFINED	**Selected Analyzer** Selected analyzer for CSR access. 0b = CSR access is pointed to Analyzer 0. 1b = CSR access is pointed to Analyzer 1.
RESERVED	31:2	0x0	RSV	N/A	Reserved.

13.2.2.10.14 Pipe Monitor Threshold - GL_TPRS_PM_THR (0x00202000; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PM_THR	13:0	0x2B20	RW	UNDEFINED	**Pipe Monitor Threshold** In byte count resolution.
RESERVED	31:14	0x0	RSV	N/A	Reserved.

13.2.2.10.15 MNG Pipe Monitor Threshold - GL_TPRS_MNG_PM_THR (0x00202004; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MNG_PM_THR	13:0	0x1	RW	UNDEFINED	**Management Pipe Monitor Threshold** Pipe monitor threshold for management commands.
RESERVED	31:14	0x0	RSV	N/A	Reserved.

13.2.2.10.16 Pipe Monitor Counters Status - GL_TPRS_PM_CNT[n] (0x00202008 + 0x4*n, n=0...1; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GL_PRS_PM_CNT	13:0	0x0	RO	UNDEFINED	**Pipe Monitor Counter** Contains the current PM Counter value.
RESERVED	31:14	0x0	RSV	N/A	Reserved.

13.2.2.10.17 Tx-Query Min/Max Size Control - GL_PRS_TX_SIZE_CTRL (0x00202014; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MIN_SIZE	9:0	0x3C	RW	UNDEFINED	**Minimum Size** Minimum frame size (bytes).
RESERVED	14:10	0x0	RSV	N/A	Reserved.
MIN_SIZE_EN	15	1b	RW	UNDEFINED	**Minimum Size Enable** Min size padding enable.
MAX_SIZE	25:16	0x1F8	RW	UNDEFINED	**Maximum Size** Maximum frame size (bytes).
RESERVED	30:26	0x0	RSV	N/A	Reserved.
MAX_SIZE_EN	31	1b	RW	UNDEFINED	**Maximum Size Enable** Max size chopping enable.

13.2.2.10.18 Tx-Query Pipe-Status Init for Word 0-6 - GL_PRS_TX_PIPE_INIT0[n] (0x00202018 + 0x4*n, n=0...6; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GPCSR_INIT	15:0	0xFFFF	RW	UNDEFINED	**GPCSR Initial** Initial value of this word in the pipe_status.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.10.19 Tx-Query Pipe-Status Init for Word 7 - GL_PRS_TX_PIPE_INIT1 (0x00202034; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GPCSR_INIT	15:0	0x1000	RW	UNDEFINED	**GPCSR Initial** Initial value of this word in the pipe_status.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.10.20 Tx-Query Pipe-Status Init for Word 8 - GL_PRS_TX_PIPE_INIT2 (0x00202038; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GPCSR_INIT	15:0	0x2492	RW	UNDEFINED	**GPCSR Initial** Initial value of this word in the pipe_status.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.10.21 Tx ANA CSR Access Control - GL_TPRS_ANA_CSR_CTRL (0x00202100; RO)

Field definitions are the same as those defined in Section 13.2.2.10.13.

13.2.2.10.22 XLR Marker Trigger PE - GL_XLR_MARKER_TRIG_PE (0x005008C0; RO)

Field definitions are the same as those defined in Section 13.2.2.10.5.

13.2.2.11 PF - Switch Registers

Registers used to describe the switch behavior.

13.2.2.11.1 Port - TC Transmit UP Replacement - PRT_TCTUPR[n] (0x00040840 + 0x4*n, n=0...31; RW)

Defines the TC-based Tx UP remapping.

Field	Bit(s)	Init.	Type	CFG Policy	Description
UP0	2:0	000b	RW	UNDEFINED	**UP 0** Defines the UP to set if UP in packet is zero and packet is sent through TC #n.
RESERVED	3	0b	RSV	N/A	Reserved.
UP1	6:4	001b	RW	UNDEFINED	**UP 1** Defines the UP to set if UP in packet is one and packet is sent through TC #n.
RESERVED	7	0b	RSV	N/A	Reserved.
UP2	10:8	010b	RW	UNDEFINED	**UP 2** Defines the UP to set if UP in packet is two and packet is sent through TC #n.
RESERVED	11	0b	RSV	N/A	Reserved.
UP3	14:12	011b	RW	UNDEFINED	**UP 3** Defines the UP to set if UP in packet is three and packet is sent through TC #n.
RESERVED	15	0b	RSV	N/A	Reserved.
UP4	18:16	100b	RW	UNDEFINED	**UP 4** Defines the UP to set if UP in packet is four and packet is sent through TC #n.
RESERVED	19	0b	RSV	N/A	Reserved.
UP5	22:20	101b	RW	UNDEFINED	**UP 5** Defines the UP to set if UP in packet is five and packet is sent through TC #n.
RESERVED	23	0b	RSV	N/A	Reserved.
UP6	26:24	110b	RW	UNDEFINED	**UP 6** Defines the UP to set if UP in packet is six and packet is sent through TC #n.
RESERVED	27	0b	RSV	N/A	Reserved.
UP7	30:28	111b	RW	UNDEFINED	**UP 7** Defines the UP to set if UP in packet is seven and packet is sent through TC #n.
RESERVED	31	0b	RSV	N/A	Reserved.

13.2.2.11.2 IPsec Function limiting - GL_SWT_FUNCFILT (0x001D2698; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FUNCFILT	0	0b	RW	UNDEFINED	**Function Filter** If set, receiving VSI function should match the function of the packet as indicated by the description engine.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.11.3 Large Action - Single Action Offset - GL_SWT_LAT_SINGLE (0x00204000; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BASE	10:0	0x0	RW	UNDEFINED	**Base** Base of Single Action section in Large Action Table (encoded as base/4).
RESERVED	15:11	0x0	RSV	N/A	Reserved.
SIZE	26:16	0x0	RW	UNDEFINED	**Size** Size of Single Action section in Large Action Table (encoded as size/4).
RESERVED	31:27	0x0	RSV	N/A	Reserved.

13.2.2.11.4 Large Action - Double Action Offset - GL_SWT_LAT_DOUBLE (0x00204004; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BASE	10:0	0x0	RW	UNDEFINED	**Base** Base of Double Action section in Large Action Table (encoded as base/4).
RESERVED	15:11	0x0	RSV	N/A	Reserved.
SIZE	26:16	0x0	RW	UNDEFINED	**Size** Size of Double Action section in Large Action Table (encoded as size/4).
RESERVED	31:27	0x0	RSV	N/A	Reserved.

13.2.2.11.5 Large Action - Quad Action Offset - GL_SWT_LAT_QUAD (0x00204008; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BASE	10:0	0x0	RW	UNDEFINED	**Base** Base of Quad Action section in Large Action Table (encoded as base/4).
RESERVED	15:11	0x0	RSV	N/A	Reserved.
SIZE	26:16	0x0	RW	UNDEFINED	**Size** Size of Quad Action section in Large Action Table (encoded as size/4).
RESERVED	31:27	0x0	RSV	N/A	Reserved.

13.2.2.11.6 Replication Table Control - EMP_SWT_REPIND (0x0020401C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
OPCODE	3:0	0x0	RW	UNDEFINED	**Opcode** 0x0 = Done 0x1 = Write (clears on done) 0x2 = Read (clears on done) 0x3 = Write All (clears on done) 0x4 = Read All “And” (clears on done) 0x5 = Read All “Nor” (clears on done) 0x6 = Reserved 0x7 = Read Error (status)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LIST_INDEX_NUMBER	13:4	0x0	RW	UNDEFINED	**List Index Number** Address of the prune table. Allows 0-1023 access range.
RESERVED	15:14	00b	RSV	N/A	Reserved.
VSI_NUM	25:16	0x0	RW	UNDEFINED	**VSI Number** Bit index of VSI bit vector (prune table line). Allowed values are 0-383. Accessing >383 causes "Read Error" status.
RESERVED	30:26	0x0	RSV	N/A	Reserved.
BIT_VALUE	31	0b	RW	UNDEFINED	**Bit Value** Write — Bit write value to VSI index. Read — Upon Opcode=0, bit read value of VSI index. Write All — Bit write value to all VSIs (entire line). Read All "And" — The "And" (&) result of entire line (all ones). Read All "Nor" — The "Nor" result of entire line (all zeros).

13.2.2.11.7 Prune Table Control - EMP_SWT_PRUNIND (0x00204020; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
OPCODE	3:0	0x0	RW	UNDEFINED	**Opcode** 0x0 = Done 0x1 = Write (clears on done) 0x2 = Read (clears on done) 0x3 = Write All (clears on done) 0x4 = Read All "And" (clears on done) 0x5 = Read All "Nor" (clears on done) 0x6 = Reserved 0x7 = Read Error (status)
LIST_INDEX_NUMBER	13:4	0x0	RW	UNDEFINED	**List Index Number** Address of the prune table. Allows 0-1023 access range.
RESERVED	15:14	00b	RSV	N/A	Reserved.
VSI_NUM	25:16	0x0	RW	UNDEFINED	**VSI Number** Bit index of VSI bit vector (prune table line). Allowed values are 0-383. Accessing >383 causes "Read Error" status.
RESERVED	30:26	0x0	RSV	N/A	Reserved.
BIT_VALUE	31	0b	RW	UNDEFINED	**Bit Value** Write — Bit write value to VSI index. Read — Upon Opcode=0, bit read value of VSI index. Write All — Bit write value to all VSIs (entire line). Read All "And" — The "And" (&).

13.2.2.11.8 Unallowed Override Attempt Count - GL_OVERRIDEC (0x002040A4; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
OVERRIDE_ATTEMPTC	15:0	0x0	RWC	UNDEFINED	**Override Attempt Count** Increments on each override attempt from source VSI for which VSI_SRCSWCTRL.*ALLOWDESTOVERRIDE* is not set (override attempt is setting non-zero value in the *SWTCH* field of the descriptor).
LAST_VSI	25:16	0x0	RWC	UNDEFINED	**Last VSI** The value of the last source VSI that attempted override and its VSI_SRCSWCTRL.*ALLOWDESTOVERRIDE* is not set (override attempt is setting non-zero value in the *SWTCH* field of the descriptor).
RESERVED	31:26	0x0	RSV	N/A	Reserved.

13.2.2.11.9 Switch Metadata Priority - GL_SWT_MD_PRI (0x002040AC; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VSI_PRI	2:0	011b	RW	UNDEFINED	**VSI Priority** Priority of VSI action.
RESERVED	3	0b	RSV	N/A	Reserved.
LB_PRI	6:4	011b	RW	UNDEFINED	**Loopback Priority** Priority of Loopback Enable action.
RESERVED	7	0b	RSV	N/A	Reserved.
LAN_EN_PRI	10:8	011b	RW	UNDEFINED	**LAN Enable Priority** Priority of LAN Enable action.
RESERVED	11	0b	RSV	N/A	Reserved.
QH_PRI	14:12	110b	RW	UNDEFINED	**Queue High Priority** Priority of queuing actions with *Q_SET* = 1 in action.
RESERVED	15	0b	RSV	N/A	Reserved.
QL_PRI	18:16	100b	RW	UNDEFINED	**Queue Low Priority** Priority of queuing actions with *Q_SET* = 0 in action.
RESERVED	31:19	0x0	RSV	N/A	Reserved.

13.2.2.11.10 Storm Control - Multicast Current Count - PRT_SWT_MSCCNT (0x00204100; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CCOUNT	24:0	0x0	RO	N/A	**Current Count** IBSC traffic current count. Represents the count of multicast traffic received in the current time interval in units of 64-byte segments.
RESERVED	31:25	0x0	RSV	N/A	Reserved.

13.2.2.11.11 Port - Store Bad Packets VSI - PRT_SBPVSI (0x00204120; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BAD_FRAMES_VSI	9:0	0x0	RW	UNDEFINED	**Bad Frames VSI** Indicates the VSI used to forward error packets if the *SBP* bit is set.
RESERVED	30:10	0x0	RSV	N/A	Reserved.
SBP	31	0b	RW	UNDEFINED	**Store Bad Packets** If set, pass bad frames to the VSI defined in the *BAD_FRAMES_VSI* field.

13.2.2.11.12 Storm Control - Status - PRT_SCSTS (0x00204140; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BSCA	0	0b	RO	N/A	**Broadcast Storm Control Active** Broadcast storm control active.
BSCAP	1	0b	RO	N/A	**Broadcast Storm Control Active Previous** Broadcast storm control active in previous window.
MSCA	2	0b	RO	N/A	**Multicast Storm Control Active** Multicast storm control active.
MSCAP	3	0b	RO	N/A	**Multicast Storm Control Active Previous** Multicast storm control active in previous window.
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.11.13 Storm Control - Broadcast Current Count - PRT_SWT_BSCCNT (0x00204160; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CCOUNT	24:0	0x0	RO	N/A	**Current Count** IBSC traffic current count. Represents the count of broadcast traffic received in the current time interval in units of 64-byte segments.
RESERVED	31:25	0x0	RSV	N/A	Reserved.

13.2.2.11.14 Storm Control - Broadcast Threshold - PRT_SWT_BSCTRH (0x00204180; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UTRESH	18:0	0x0	RW	UNDEFINED	**Upper Threshold** Traffic upper threshold size. Represents the upper threshold for broadcast storm control.
RESERVED	31:19	0x0	RSV	N/A	Reserved.

13.2.2.11.15 Storm Control - Multicast Threshold - PRT_SWT_MSCTRH (0x002041C0; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UTRESH	18:0	0x0	RW	UNDEFINED	**Upper Threshold** Traffic upper threshold size. Represents the upper threshold for multicast storm control.
RESERVED	31:19	0x0	RSV	N/A	Reserved.

13.2.2.11.16 Storm Control - Basic Interval - PRT_SWT_SCBI (0x002041E0; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BI	24:0	0x2710	RW	UNDEFINED	**Basic Interval** Basic interval in 100 Mb/s port link rate in 1 μs strobes. The default is equivalent to 1 Mbit of data. ***Notes:*** 1. Initial value defines a basic interval of 10 ms at 100 Mb/s link speed. 2. The interval in higher port link rates is divided by the link speed/100 Mb/s to get the new interval.
RESERVED	31:25	0x0	RSV	N/A	Reserved.

13.2.2.11.17 Storm Control - Control Register - PRT_SWT_SCCRL (0x00204200; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MDIPW	0	0b	RW	UNDEFINED	**Multicast Drop if Previous Window** Drop multicast packets (excluding flow control packets) if multicast threshold is exceeded in previous window.
MDICW	1	0b	RW	UNDEFINED	**Multicast Drop if Current Window** Drop multicast packets (excluding flow control packets) if multicast threshold is exceeded in current window.
BDIPW	2	0b	RW	UNDEFINED	**Broadcast Drop if Previous Window** Drop broadcast packets if broadcast threshold is exceeded in previous window.
BDICW	3	0b	RW	UNDEFINED	**Broadcast Drop if Current Window** Drop broadcast packets if broadcast threshold is exceeded in current window.
RESERVED	7:4	0x0	RSV	N/A	Reserved.
INTERVAL	27:8	0xA	RW	UNDEFINED	**Interval** BSC/MSC time interval specification. The interval size for applying Ingress Broadcast or Multicast Storm Control. Interrupt decisions are made at the end of each interval (and most flags are also set at interval end). Setting this field resets the counter.
RESERVED	31:28	0x0	RSV	N/A	Reserved.

13.2.2.11.18 Mirror - LAN Port Ingress Rule - PRT_SWT_MIRIG (0x00204280; RO)

This register defines the rules that request to mirror ingress VSI n.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MIRRULE	5:0	0x0	RW	UNDEFINED	**Mirror Rule** Mirror rule index.
RESERVED	6	0b	RSV	N/A	Reserved.
MIRENA	7	0b	RW	UNDEFINED	**Mirror Enable** Mirror enable.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.11.19 Mirror - LAN Port Egress Rule - PRT_SWT_MIREG (0x002042A0; RO)

This register defines the rules that request to mirror egress VSI n.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MIRRULE	5:0	0x0	RW	UNDEFINED	**Mirror Rule** Mirror rule index.
RESERVED	6	0b	RSV	N/A	Reserved.
MIRENA	7	0b	RW	UNDEFINED	**Mirror Enable** Mirror enable.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.11.20 Mirror - Target VSI - GL_SWT_MIRTARVSI[n] (0x00204500 + 0x4*n, n=0...63; RO)

This register defines the destination VSI of mirror rules.

Field	Bit(s)	Init.	Type	CFG Policy	Description
VFVMNUMBER	9:0	0x0	RW	UNDEFINED	**VF/VM Number** Defines the VF this VSI belongs to, or the VM within the PF associated with this VSI. Must be zero if VSI type is PF or EMP (VSI_VSI2F.*FUNCTIONTYPE* =10b or 11b, respectively). If the VSI is of type VM (VSI_VSI2F.*FUNCTIONTYPE* = 01b), this number should be equal to the VSI number.
FUNCTIONTYPE	11:10	00b	RW	UNDEFINED	**Function Type** Defines the type of VSI: 00b = VF 01b = VM 10b = PF 11b = EMP
PFNUMBER	14:12	000b	RW	UNDEFINED	**PF Number** Defines the PF this VSI belongs to. Should be set to the PF number for all VSI types except EMP (VSI_VSI2PF.*FUNCTIONTYPE* = 00b, 01b, or 10b).
RESERVED	19:15	0x0	RSV	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
TARGETVSI	29:20	0x0	RW	UNDEFINED	**Target VSI** Defines the target VSI of mirror rule #n.
RESERVED	30	0b	RSV	N/A	Reserved.
RULEENABLE	31	0b	RW	UNDEFINED	**Rule Enable** If set, this mirror rule is enabled.

13.2.2.11.21 SWID Stat Block ID - GLSWID_STAT_BLOCK[n] (0x0020A1A4 + 0x4*n, n=0...255; RO)

Defines if one of 32 VEB statistic block is associated with this switch ID.

Field	Bit(s)	Init.	Type	CFG Policy	Description
VEBID	4:0	0x0	RW	UNDEFINED	**VEB ID** Defines the stat block ID to which the switch ID is associated.
RESERVED	30:5	0x0	RSV	N/A	Reserved.
VEBID_VALID	31	0b	RW	UNDEFINED	**VEB ID Valid** A switch ID may not be associated with any switch block (GLSWID_STAT_BLOCK.*VEBID_VALID* = 0b)

13.2.2.11.22 Switch Recipes Used - GLSWT_ACT_RESP_0 (0x0020A5A4; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GLSWT_ACT_RESP	31:0	0x0	RO	UNDEFINED	**Action Recipes** Bitmap of recipes used in actions.

13.2.2.11.23 Switch Recipes Used - GLSWT_ACT_RESP_1 (0x0020A5A8; RO)

Field definitions are the same as those defined in Section 13.2.2.11.22.

13.2.2.11.24 Throughput Counters Config - GL_PLG_AVG_CALC_CFG (0x0020A5AC; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CYCLE_LEN	30:0	0x0	RW	UNDEFINED	**Cycle Length**
MODE	31	0b	RW	UNDEFINED	**Mode**

13.2.2.11.25 Throughput Counters Status - GL_PLG_AVG_CALC_ST (0x0020A5B0; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IN_DATA	14:0	0x0	RO	UNDEFINED	**In Data**
RESERVED	15	0b	RSV	N/A	Reserved.
OUT_DATA	30:16	0x0	RO	UNDEFINED	**Out Data**
VALID	31	0b	RO	UNDEFINED	**Valid**

13.2.2.11.26 Hardware Arb Control - GLSWT_ARB_MODE (0x0020A674; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FLU_PRI_SHM	0	0b	RW	UNDEFINED	**FLU Priority** rcu_swr FLU-->KGG collectors ARB: 0b = Prioritized according to FIFO level (above occupancy of 4). 1b = RR.
TX_RX_FWD_PRI	1	0b	RW	UNDEFINED	**Tx/Rx Forward Priority** Tx/Rx replica arbitration at MFIFO output: 0b = RR. 1b = SP to Tx.
RESERVED	31:2	0x0	RSV	N/A	Reserved.

13.2.2.11.27 Recipe Data - GL_PRE_CFG_DATA[n] (0x00214074 + 0x4*n, n=0...6; RO)

Indirect configuration of RCU pre-LUTs.

Field	Bit(s)	Init.	Type	CFG Policy	Description
GL_PRE_RCP_DATA	31:0	0x0	RW	UNDEFINED	**Pre RCP Data**

13.2.2.11.28 Recipe Command - GL_PRE_CFG_CMD (0x00214090; RO)

Indirect configuration of RCU pre-LUTs.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ADDR	12:0	0x0	WO	UNDEFINED	**Address** Index within the table.
RESERVED	15:13	000b	RSV	N/A	Reserved.
TBLIDX	18:16	000b	WO	UNDEFINED	**Table Index** 000b = Recipes 001b = Profile to recipe mapping 010b = Queue context 011b = Large actions 100b = Dependent recipes table All other values are reserved.
RESERVED	28:19	0x0	RSV	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
CMD	29	0b	WO	UNDEFINED	**Command** 0b = Read 1b = Write
RESERVED	30	0b	RSV	N/A	Reserved.
DONE	31	0b	ROCV	N/A	**Done** Done indication. Cleared on read.

13.2.2.11.29 Virtual Port Switch ID - GL_VP_SWITCHID[n] (0x00214094 + 0x4*n, n=0...31; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SWITCHID	7:0	0x0	RW	UNDEFINED	**Switch ID** Source VSI Switch ID.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.11.30 SWID Field Vector Index - GL_SWT_SWIDFVIDX (0x00214114; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SWIDFVIDX	5:0	0x0	RW	UNDEFINED	**Switch ID Field Vector Index** Destination of Switch ID in the Field Vector.
RESERVED	30:6	0x0	RSV	N/A	Reserved.
PORT_TYPE	31	0b	RW	UNDEFINED	**Port Type** Selects Virtual or Physical port Switch ID. 0b = Physical 1b = Virtual

13.2.2.11.31 FW Config Status - GL_SWT_FW_STS[n] (0x00216000 + 0x4*n, n=0...5; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GL_SWT_FW_STS	31:0	0x0	RO	UNDEFINED	**Firmware Config Status**

13.2.2.12 PF - VSI Context Registers

These registers create the VSI context. Unless otherwise stated, these registers are accessed by internal firmware. There are 768 VSI contexts shared between all PFs. All VSIs are accessible to all functions, but each function should access only the VSIs allocated to it.

13.2.2.12.1 VSI Tag Insert Register - First Tag - VSI_TIR_0[VSI] (0x00041000 + 0x4*VSI, VSI=0...767; RO)

Controls the port (VSI) based insertion of tags.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PORT_TAG_ID	15:0	0x0	RW	UNDEFINED	**Port Tag ID** Port (VSI) L2 tag to insert. The tag to insert to is defined in the VSI_L2TAGSTXVALID.*TIR0INSERTID* field. The insert enable is VSI_L2TAGSTXVALID.*TIR0_INSERT*.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.12.2 VSI Tag Insert Register - Second Tag - VSI_TIR_1[VSI] (0x00042000 + 0x4*VSI, VSI=0...767; RO)

Controls the port (VSI) based insertion of tags.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PORT_TAG_ID	31:0	0x0	RW	UNDEFINED	**Port Tag ID** Port (VSI) L2 tag to insert. The tag to insert to is defined in the VSI_L2TAGSTXVALID.*TIR1INSERTID* field. The insert enable is VSI_L2TAGSTXVALID.*TIR1_INSERT*.

13.2.2.12.3 VSI Tag Insert Register - Third Tag - VSI_TIR_2[VSI] (0x00043000 + 0x4*VSI, VSI=0...767; RO)

Controls the port (VSI) based insertion of tags.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PORT_TAG_ID	15:0	0x0	RW	UNDEFINED	**Port Tag ID** Port (VSI) L2 tag to insert. The tag to insert to is defined in the VSI_L2TAGSTXVALID.*TIR2INSERTID* field. The insert enable is VSI_L2TAGSVALID.*TIR2_INSERT*.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.12.4 VSI Tag Alternate Insert Register - VSI_TAIR[VSI] (0x00044000 + 0x4*VSI, VSI=0...767; RO)

Contains the alternate port (VSI) based tag for selected queues.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PORT_TAG_ID	15:0	0x0	RW	UNDEFINED	**Port Tag ID** Port VLAN tag to insert if the VSI_L2TAGSTXVALID.*TIR0_INSERT* is set and the *ALT_VLAN* bit in the transmit queue context is set.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.12.5 VSI Tag Accept Register - VSI_TAR[VSI] (0x00045000 + 0x4*VSI, VSI=0...767; RO)

Defines which tags can be accepted from the driver. This register impacts only tag insertion from the driver and not tag insertion by the hardware.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ACCEPTTAGGED	9:0	0x0	RW	UNDEFINED	**Accept Tagged** A bitmap describing if a packet with tag N is accepted. ***Note:*** The two first bits are mapped to the pre L2 and MAC headers.
RESERVED	15:10	0x0	RSV	N/A	Reserved.
ACCEPTUNTAGGED	25:16	0x0	RW	UNDEFINED	**Accept Untagged** A bitmap describing if a packet without tag N is accepted (if L2TAGCTRL.*ISVLAN* is set, admits also priority tagged packets (VLAN tag = 0)). ***Note:*** The two first bits are mapped to the pre L2 and MAC headers.
RESERVED	31:26	0x0	RSV	N/A	Reserved.

13.2.2.12.6 VSI L2 Tx Tags Control - VSI_L2TAGSTXVALID[VSI] (0x00046000 + 0x4*VSI, VSI=0...767; RO)

Identifies which of the eight L2 tags are available for this port for Tx traffic.

Note: The IDs for all the valid tags should be different.

Field	Bit(s)	Init.	Type	CFG Policy	Description
L2TAG1INSERTID	2:0	000b	RW	UNDEFINED	**L2 Tag1 Insert ID** Defines the tag ID to which the value in the L2TAG1 descriptor field should be inserted. This field is valid only if *L2TAG1INSERTID_VALID* is set. Otherwise, insertion requests from L2TAG1 are ignored.
L2TAG1INSERTID_VALID	3	0b	RW	UNDEFINED	**L2 Tag1 Inserted ID Valid** If set, the *L2TAG1INSERTID* field contains a valid value.
L2TAG2INSERTID	6:4	000b	RW	UNDEFINED	**L2 Tag2 Insert ID** Defines the tag ID to which the value in the L2TAG2 descriptor field should be inserted. This field is valid only if *L2TAG2INSERTID_VALID* is set. Otherwise, insertion requests from L2TAG2 are ignored.
L2TAG2INSERTID_VALID	7	0b	RW	UNDEFINED	**L2 Tag2 Inserted ID Valid** If set, the *L2TAG2INSERTID* field contains a valid value.
RESERVED	15:8	0x0	RSV	N/A	Reserved.
TIR0INSERTID	18:16	000b	RW	UNDEFINED	**TIR0 Insert ID** Defines the tag ID to which the value in the VSI_TIR[0].*PORT_TAG_ID* field should be inserted. The tags encoding is described in Section 7.12.3.4, "Tag Handling - Programming Interface". The tag is inserted only if the *TIR0_INSERT* bit is set. This tag is the only one that supports alternate tagging using the VSI_TAIR value.

Field	Bit(s)	Init.	Type	CFG Policy	Description
TIR0_INSERT	19	0b	RW	UNDEFINED	**TIR0 Insert** Insert the tag in VSI_TIR_0 to the L2 Tag pointed by the *TIR0INSERID* field.
TIR1INSERTID	22:20	000b	RW	UNDEFINED	**TIR1 Insert ID** Defines the tag ID to which the value in the VSI_TIR[1].*PORT_TAG_ID* field should be inserted. See *TIR0INSERTID* description for this field encoding. The tag is inserted only if the *TIR1_INSERT* bit is set.
TIR1_INSERT	23	0b	RW	UNDEFINED	**TIR1 Insert** Insert the tag in VSI_TIR_1 to the L2 Tag pointed by the *TIR1INSERID* field.
TIR2INSERTID	26:24	000b	RW	UNDEFINED	**TIR2 Insert ID** Defines the tag ID to which the value in the VSI_TIR[2].*PORT_TAG_ID* field should be inserted. See *TIR0INSERTID* description for this field encoding. The tag is inserted only if the *TIR2_INSERT* bit is set.
TIR2_INSERT	27	0b	RW	UNDEFINED	**TIR2 Insert** Insert the tag in VSI_TIR_2 to the L2 Tag pointed by the *TIR2INSERID* field.
RESERVED	31:28	0x0	RSV	N/A	Reserved.

13.2.2.12.7 VSI Transmit UP Replacement - VSI_TUPR[VSI] (0x00047000 + 0x4*VSI, VSI=0...767; RO)

Defines the VSI-based Tx UP remapping.

Field	Bit(s)	Init.	Type	CFG Policy	Description
UP0	2:0	000b	RW	UNDEFINED	**UP 0** Defines the UP to set if UP in packet is zero.
UP1	5:3	000b	RW	UNDEFINED	**UP 1** Defines the UP to set if UP in packet is one.
UP2	8:6	000b	RW	UNDEFINED	**UP 2** Defines the UP to set if UP in packet is two.
UP3	11:9	000b	RW	UNDEFINED	**UP 3** Defines the UP to set if UP in packet is three.
UP4	14:12	000b	RW	UNDEFINED	**UP 4** Defines the UP to set if UP in packet is four.
UP5	17:15	000b	RW	UNDEFINED	**UP 5** Defines the UP to set if UP in packet is five.
UP6	20:18	000b	RW	UNDEFINED	**UP 6** Defines the UP to set if UP in packet is six.
UP7	23:21	000b	RW	UNDEFINED	**UP 7** Defines the UP to set if UP in packet is seven.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.12.8 VSI Transmit UP Inner to Outer Mapping - VSI_TUPIOM[VSI] (0x00048000 + 0x4*VSI, VSI=0...767; RO)

Defines the VSI-based Tx UP remapping.

Field	Bit(s)	Init.	Type	CFG Policy	Description
UP0	2:0	000b	RW	UNDEFINED	**UP 0** Defines the UP to set in outer tag if UP in inner tag is zero.
UP1	5:3	000b	RW	UNDEFINED	**UP 1** Defines the UP to set in outer tag if UP in inner tag is one.
UP2	8:6	000b	RW	UNDEFINED	**UP 2** Defines the UP to set in outer tag if UP in inner tag is two.
UP3	11:9	000b	RW	UNDEFINED	**UP 3** Defines the UP to set in outer tag if UP in inner tag is three.
UP4	14:12	000b	RW	UNDEFINED	**UP 4** Defines the UP to set in outer tag if UP in inner tag is four.
UP5	17:15	000b	RW	UNDEFINED	**UP 5** Defines the UP to set in outer tag if UP in inner tag is five.
UP6	20:18	000b	RW	UNDEFINED	**UP 6** Defines the UP to set in outer tag if UP in inner tag is six.
UP7	23:21	000b	RW	UNDEFINED	**UP 7** Defines the UP to set in outer tag if UP in inner tag is seven.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.12.9 VSI Receive UP Replacement - VSI_RUPR[VSI] (0x00050000 + 0x4*VSI, VSI=0...767; RW)

Defines the VSI based Rx UP remapping. The tag on which this translation is done is the tag for which the L2TAGSCTRL.*INNERUP* is set.

Field	Bit(s)	Init.	Type	CFG Policy	Description
UP0	2:0	000b	RW	UNDEFINED	**UP 0** Defines the UP to set if UP in packet is zero.
UP1	5:3	000b	RW	UNDEFINED	**UP 1** Defines the UP to set if UP in packet is one.
UP2	8:6	000b	RW	UNDEFINED	**UP 2** Defines the UP to set if UP in packet is two.
UP3	11:9	000b	RW	UNDEFINED	**UP 3** Defines the UP to set if UP in packet is three.
UP4	14:12	000b	RW	UNDEFINED	**UP 4** Defines the UP to set if UP in packet is four.
UP5	17:15	000b	RW	UNDEFINED	**UP 5** Defines the UP to set if UP in packet is five.
UP6	20:18	000b	RW	UNDEFINED	**UP 6** Defines the UP to set if UP in packet is six.
UP7	23:21	000b	RW	UNDEFINED	**UP 7** Defines the UP to set if UP in packet is seven.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.12.10 VSI Tag Strip Register - VSI_TSR[VSI] (0x00051000 + 0x4*VSI, VSI=0...767; RW)

Defines the behavior of tag extraction from receive packets.

Field	Bit(s)	Init.	Type	CFG Policy	Description
STRIPTAG	9:0	0x0	RW	UNDEFINED	**Strip Tag** A per tag bitmap defining which tags to strip from the packet. The *SHOWTAG* and *SHOWPRIONLY* fields define which part of the tag to extract to descriptor. ***Note:*** The two LS bits of this field relate to the *pre*-L2 and MAC headers and should not be set. Bits 9:2 should be used to configure the striping of the L2 tags.
SHOWTAG	19:10	0x0	RW	UNDEFINED	**Show Tag** A per tag bitmap defining which tags to extract to the descriptor. The *SHOWPRIONLY* field defines which part of the tag to extract to the descriptor. At most, two of these bits should be set. If more than two bits are set, only the two first ones are considered. See For details on the format in the descriptor, see Section 10.4.2.2, "Receive Descriptor - Write-Back Format".
SHOWPRIONLY	29:20	0x0	RW	UNDEFINED	**Strip Priority Only** A per tag bitmap defining which par of the tags to extract to the descriptor. If set, only the priority bits are extracted. Otherwise the entire tag is used. Relevant only if the corresponding bit in SHOWTAG is set. ***Note:*** The two LS bits of this field relate to the pre-L2 and MAC headers and should not be set. Bits 9:2 should be used to configure the exposure of the L2 tags priority bits.
RESERVED	31:30	00b	RSV	N/A	Reserved.

13.2.2.12.11 PASID Context - VSI_PASID[VSI] (0x0009C000 + 0x4*VSI, VSI=0...767; RW)

PASID context per VSI.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PASID	19:0	0x0	RW	UNDEFINED	**Process Address Space ID** PASID value of PASID TLP prefix.
RESERVED	30:20	0x0	RSV	N/A	Reserved.
EN	31	0b	RW	UNDEFINED	**Enable** Valid for *PASID* field.

13.2.2.12.12 VSI to Function Mapping Multicast - VSI_VSI2F[VSI] (0x001D0000 + 0x4*VSI, VSI=0...767; RW)

This register indicates the function owning the VSI.

Field	Bit(s)	Init.	Type	CFG Policy	Description
VFVMNUMBER	9:0	0x0	RW	UNDEFINED	**VF/VM Number** Defines the VF this VSI belongs to, or the VM within the PF associated with this VSI. Must be zero if VSI type is PF or EMP (VSI_VSI2F.*FUNCTIONTYPE* =10b or 11b, respectively). If the VSI is of type VM (VSI_VSI2F.*FUNCTIONTYPE* = 01b), this number should be equal to the VSI number.
FUNCTIONTYPE	11:10	00b	RW	UNDEFINED	**Function Type** Defines the type of VSI: 00b = VF 01b = VM 10b = PF 11b = EMP
PFNUMBER	14:12	000b	RW	UNDEFINED	**PF Number** Defines the PF this VSI belongs to. Should be set to the PF number for all VSI types except EMP (VSI_VSI2PF.*FUNCTIONTYPE* = 00b, 01b, or 10b).
RESERVED	15	0b	RSV	N/A	Reserved.
BUFFERNUMBER	18:16	0x0	RW	UNDEFINED	**Buffer Number** Defines the buffer in the EMP that is used for this VSI. Relevant only if *FUNCTIONTYPE* = EMP (11b).
RESERVED	19	0b	RSV	N/A	Reserved.
VSI_NUMBER	29:20	0x0	RW	UNDEFINED	**VSI Number** Defines the VSI number of this VSI. The default of this field in the VSI#. This field is not used for VSI redirection. ***Note:*** This field is a candidate for removal in future gen products and software/firmware should refrain from referring it.
RESERVED	30	0b	RSV	N/A	Reserved.
VSI_ENABLE	31	0b	RW	UNDEFINED	**VSI Enable** When set, this VSI is enabled and can receive or transmit packets.

13.2.2.12.13 VSI Rx Switch Control - VSI_RXSWCTRL[VSI] (0x00205000 + 0x4*VSI, VSI=0...767; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	7:0	0x0	RSV	N/A	Reserved.
MACVSIPRUNEENABLE	8	0b	RW	UNDEFINED	**MAC VSI Prune Enable** If set, inverse actions are applied to Rx traffic to this VSI (used in default image for MAC source pruning).
PRUNEENABLE	12:9	0x0	RW	UNDEFINED	**Prune Enable** Defines which pruning modes are enabled for Rx traffic. Each bit matches one of the prune index in the recipe (offset 181:180).
SRCPRUNEENABLE	13	0b	RW	UNDEFINED	**Source Prune Enable** Enable source pruning. If set, a packet is not received by this VSI if sent from it.
RESERVED	31:14	0x0	RSV	N/A	Reserved.

13.2.2.12.14 Mirror - Rx Rules VSIs - VSI_SWT_MIREG[VSI] (0x00207000 + 0x4*VSI, VSI=0...767; RO)

This register defines the rules that request to mirror egress VSI n.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MIRRULE	5:0	0x0	RW	UNDEFINED	**Mirror Rule** Mirror rule index.
RESERVED	6	0b	RSV	N/A	Reserved.
MIRENA	7	0b	RW	UNDEFINED	**Mirror Enable** Mirror enable.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.12.15 Mirror - Tx Rules VSIs - VSI_SWT_MIRIG[VSI] (0x00208000 + 0x4*VSI, VSI=0...767; RO)

This register defines the rules that request to mirror ingress VSI n.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MIRRULE	5:0	0x0	RW	UNDEFINED	**Mirror Rule** Mirror rule index.
RESERVED	6	0b	RSV	N/A	Reserved.
MIRENA	7	0b	RW	UNDEFINED	**Mirror Enable** Mirror enable.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.12.16 VSI Source Switch Control - VSI_SRCSWCTRL[VSI] (0x00209000 + 0x4*VSI, VSI=0...767; RO)

These registers contain the switch ID for each VF. If the VF is identified by an STag, this field should include the STag and the *ISNSTAG* should be cleared. Otherwise, it should include a switch ID and the *ISNSTAG* should be set.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ALLOWDESTOVERRIDE	0	0b	RW	UNDEFINED	**Allow Destination Override** Allows destination override by transmit descriptor.
ALLOWLOOPBACK	1	0b	RW	UNDEFINED	**Allow Loopback** Allows forwarding of packets from this VSI to local VSIs.
LANENABLE	2	0b	RW	UNDEFINED	**LAN Enable** If set, packets sent from this VSI can be sent to the external port.
MACAS	3	0b	RW	UNDEFINED	**MAC Anti-Spoofing** Enabling recipes where the inverse_action is set for Tx packets (Enables MAC anti-spoofing in default setting)
PRUNEENABLE	7:4	0x0	RW	UNDEFINED	**Prune Enable** Enables prune actions for Tx traffic. According to the prune_index in recipe (offset 181:180).
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.12.17 Source VSI Switch ID - VSI_SWITCHID[VSI] (0x00215000 + 0x4*VSI, VSI=0...767; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SWITCHID	7:0	0x0	RW	UNDEFINED	**Switch ID** Source VSI Switch ID.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.12.18 VSI Classification Filter - Hash Key - VSIQF_HKEY[n,VSI] (0x00400000 + 0x1000*n + 0x4*VSI, n=0...12, VSI=0...767; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
KEY_0	7:0	0x0	RW	UNDEFINED	**Key 0** Toeplitz key byte "4*m+3" of VSI "n". Register index is "1024*m + n", where "m" range is 0-12 (total 52 key bytes), and "n" range is 0-767. ***Note:*** Index range 768-1023 of each key-quad register is unused (reserved).
KEY_1	15:8	0x0	RW	UNDEFINED	**Key 1** Toeplitz key byte "4*m+3" of VSI "n". Register index is "1024*m + n", where "m" range is 0-12 (total 52 key bytes), and "n" range is 0-767. ***Note:*** Index range 768-1023 of each key-quad register is unused (reserved).
KEY_2	23:16	0x0	RW	UNDEFINED	**Key 2** Toeplitz key byte "4*m+3" of VSI "n". Register index is "1024*m + n", where "m" range is 0-12 (total 52 key bytes), and "n" range is 0-767. ***Note:*** Index range 768-1023 of each key-quad register is unused (reserved).
KEY_3	31:24	0x0	RW	UNDEFINED	**Key 3** Toeplitz key byte "4*m+3" of VSI "n". Register index is "1024*m + n", where "m" range is 0-12 (total 52 key bytes), and "n" range is 0-767. ***Note:*** Index range 768-1023 of each key-quad register is unused (reserved).

13.2.2.12.19 VSI Classification Filter - Hash Control - VSIQF_HASH_CTL[VSI] (0x0040D000 + 0x4*VSI, VSI=0...767; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
HASH_LUT_SEL	1:0	00b	RW	UNDEFINED	**Hash LUT Select** Selects the Hash LUT for the VSI: 00b = VSI LUT 01b = Reserved 10b = PF LUT 11b = Global LUT (one of 16 Global LUTs)
GLOB_LUT	5:2	0x0	RW	UNDEFINED	**Global LUT** The Global LUT index for the VSI. This field is relevant only if the *RSS_LUT_SEL* equals to Global LUT. Otherwise it must be set to zero.

Field	Bit(s)	Init.	Type	CFG Policy	Description
HASH_SCHEME	7:6	00b	RW	UNDEFINED	**Hash Scheme** The hash type that can be one of the following: 00b = Toeplitz Hash 01b = Symmetric Toeplitz 10b = Simple XOR 11b = Reserved
TC_OVER_SEL	12:8	0x0	RW	UNDEFINED	**TC Override Select** An index to one of 32 global tables used to override TC regions of queues by some of the packet profiles defined by the GLQF_PROF2TC tables. This field is meaningful only if the *TC_OVER_ENA* flag is set. Otherwise, this field must be set to zero.
RESERVED	14:13	00b	RSV	N/A	Reserved.
TC_OVER_ENA	15	0b	RW	UNDEFINED	**TC Override Enable** Enable packet profile override the TC region of queues according to the *TC_OVER_SEL* parameter.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.12.20 VSI Classification Filter - FD Control 1 - VSIQF_FD_CTL1[VSI] (0x00411000 + 0x4*VSI, VSI=0...767; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FLT_ENA	0	0b	RW	UNDEFINED	**Filtering Enable** Enable FD filtering for the VSI.
CFG_ENA	1	0b	RW	UNDEFINED	**Configuration Enable** Enable FD programming for the VSI.
EVICT_ENA	2	0b	RW	UNDEFINED	**Evict Enable** Enable auto-evict by FIN/RST for the VSI.
RESERVED	31:3	0x0	RSV	N/A	Reserved.

13.2.2.12.21 VSI Classification Filter - PE Control 1 - VSIQF_PE_CTL1[VSI] (0x00414000 + 0x4*VSI, VSI=0...767; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PE_FLTENA	0	0b	RW	UNDEFINED	**PE Filter Enable** Enable PE filter for the VSI.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.12.22 VSI Classification Filter - Hash LUT - VSIQF_HLUT[n,VSI] (0x00420000 + 0x1000*n + 0x4*VSI, n=0...15, VSI=0...767; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LUT0	3:0	0x0	RW	UNDEFINED	**LUT 0** Hash redirection LUT entry 4 x n, where "n" is the register index.
RESERVED	7:4	0x0	RSV	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
LUT1	11:8	0x0	RW	UNDEFINED	**LUT 1** Hash redirection LUT entry 4 x n+1, where "n" is the register index.
RESERVED	15:12	0x0	RSV	N/A	Reserved.
LUT2	19:16	0x0	RW	UNDEFINED	**LUT 2** Hash redirection LUT entry 4 x n+2, where "n" is the register index.
RESERVED	23:20	0x0	RSV	N/A	Reserved.
LUT3	27:24	0x0	RW	UNDEFINED	**LUT 3** Hash redirection LUT entry 4 x n+3, where "n" is the register index.
RESERVED	31:28	0x0	RSV	N/A	Reserved.

13.2.2.12.23 VSI Classification Filter - Receive TC Queue Regions - VSIQF_TC_REGION[n,VSI] (0x00448000 + 0x1000*n + 0x4*VSI, n=0...3, VSI=0...767; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TC_BASE0	10:0	0x0	RW	UNDEFINED	**TC Base 0** TC_BASE0 in register "n" defines the TC[2*n] region base (within the VSI space). The hardware does not check if the queue index exceeds the VSI range. So it is the PF software responsibility to ensure that TC_BASE + TC_SIZE does not exceed the VSI space.
TC_SIZE0	14:11	0x0	RW	UNDEFINED	**TC Size 0** TC_SIZE0 in register "n" defines the TC[2*n] region size as follows: Values 0...8 are mapped to the following respective TC sizes: 1, 2, 4, 8, 16, 32, 64, 128, and 256, respectively. Other values (larger than 8) are not defined and should not be used or programmed.
RESERVED	15	0b	RSV	N/A	Reserved.
TC_BASE1	26:16	0x0	RW	UNDEFINED	**TC Base 1** TC_BASE1 in register "n" defines the TC[2*n+1] region base (within the VSI space). The hardware does not check if the queue index exceeds the VSI range. So it is the PF software responsibility to make sure that TC_BASE + TC_SIZE does not exceed the VSI space.
TC_SIZE1	30:27	0x0	RW	UNDEFINED	**TC Size 1** TC_SIZE1 in register "n" defines the TC[2*n+1] region size as follows: Values 0...8 are mapped to the following respective TC sizes: 1, 2, 4, 8, 16, 32, 64, 128, and 256, respectively. Other values (larger than 8) are not defined and should not be used or programmed.
RESERVED	31	0b	RSV	N/A	Reserved.

13.2.2.12.24 VSI Classification Filter - FD Default Action - VSIQF_FD_DFLT[VSI] (0x00457000 + 0x4*VSI, VSI=0...767; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DEFLT_QINDX	10:0	0x0	RW	UNDEFINED	**Default Queue Index** This field defines the receive queue index within the VSI for packets that miss the FD filter. Permitted values are in the range of the queues of the VSI.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	11	0b	RSV	N/A	Reserved.
DEFLT_TOQUEUE	14:12	000b	RW	UNDEFINED	**Default ToQueue** The ToQueue parameter associates a target queue or a queue group to the FD filter: 000b = The filter assigns a target queue defined by the QINDX. 001b-111b = The filter assigns a region of queues. The region base equals the QINDX and the region size equals 2^(ToQueue).
RESERVED	15	0b	RSV	N/A	Reserved.
COMP_QINDX	26:16	0x0	RW	UNDEFINED	**Completion Queue Index** This field defines the receive queue index of the programming VSI on which the FD filter completion status is reported when the *COMP_QUEUE* flag in the programming descriptor is set to "1".
RESERVED	27	0b	RSV	N/A	Reserved.
DEFLT_QINDX_PRIO	30:28	000b	RW	UNDEFINED	**Default Queue Index Priority** The priority of the default QINDX action.
DEFLT_DROP	31	0b	RW	UNDEFINED	**Default Drop** When set to 1b, packets that miss the FD filters are dropped.

13.2.2.12.25 VSI Classification Filter - FD VSI Space Sizes - VSIQF_FD_SIZE[VSI] (0x00462000 + 0x4*VSI, VSI=0...767; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FD_GSIZE	13:0	0x0	RW	UNDEFINED	**FD Guaranteed Size** VSI guaranteed size in the FD table.
RESERVED	15:14	00b	RSV	N/A	Reserved.
FD_BSIZE	29:16	0x0	RW	UNDEFINED	**FD Best Size** VSI best effort size in the FD table.
RESERVED	31:30	00b	RSV	N/A	Reserved.

13.2.2.12.26 VSI Classification Filter - FD VSI Space Counters - VSIQF_FD_CNT[VSI] (0x00464000 + 0x4*VSI, VSI=0...767; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FD_GCNT	13:0	0x0	RW	UNDEFINED	**FD Guaranteed Counter** VSI guaranteed filter counter in the FD table.
RESERVED	15:14	00b	RSV	N/A	Reserved.
FD_BCNT	29:16	0x0	RW	UNDEFINED	**FD Best Counter** VSI best effort filter counter in the FD table.
RESERVED	31:30	00b	RSV	N/A	Reserved.

13.2.2.13 PF - ACL Registers

13.2.2.13.1 Configuration Access Command - GL_ACL_ACCESS_CMD (0x00391000; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TABLE_ID	7:0	0x0	RW	UNDEFINED	**Table ID** TCAM/Action memory index used for the access (when relevant). Inapplicable indexes will cause the command to fail.
ENTRY_INDEX	19:8	0x0	RW	UNDEFINED	**Entry Index** Entry/Scenario/Profile index. Inapplicable indexes will cause the command to fail.
OPERATION	20	0b	RW	UNDEFINED	**Operation** 0b = Read an entry from the index specified in *ENTRY_INDEX* in the table whose index is specified in *TABLE_ID*. 1b = Write an entry to the index specified in *ENTRY_INDEX* in table whose index is specified in *TABLE_ID*.
RESERVED	23:21	000b	RSV	N/A	Reserved.
OBJ_TYPE	27:24	0x0	RW	UNDEFINED	**Object Type** Accessed object type: 0x0 = TCAM entry. 0x1 = Action memory entry. 0x2 = Scenario configuration. 0x3 = Scenario action memory configuration. 0x4 = Profile dependent configuration (except for range checkers). 0x5 = Range checker configuration (profile dependent). All other values are reserved.
RESERVED	30:28	000b	RSV	N/A	Reserved.
EXECUTE	31	0b	SC	UNDEFINED	**Execute** Execute bit. Setting this self-clearing bit initiates the operation. The hardware clears this bit one cycle after it was set.

13.2.2.13.2 Configuration Access Status - GL_ACL_ACCESS_STATUS (0x00391004; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BUSY	0	0b	RO	N/A	**Busy** Busy indication.
DONE	1	0b	RO	N/A	**Done** Done indication.
ERROR	2	0b	RO	N/A	**Error** Error indication.
OPERATION	3	0b	RO	N/A	**Operation** Operation used in last access.
ERROR_CODE	7:4	0x0	RO	N/A	**Error Code** Error code: 0x0 = Table ID out of bounds. 0x1 = Entry index out of bounds. All other values are reserved.
TABLE_ID	15:8	0x0	RO	N/A	**Table ID** Table ID used for last access.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ENTRY_INDEX	27:16	0x0	RO	N/A	**Entry Index** Entry index used for last access.
OBJ_TYPE	31:28	0x0	RO	N/A	**Object Type** Object type used for last access.

13.2.2.13.3 Byte and Word Selection Bases Select per Profile - GL_ACL_PROFILE_BWSB_SEL[n] (0x00391008 + 0x4*n, n=0...31; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BSB_SRC_OFF	5:0	0x0	RW	UNDEFINED	**Source Byte Offset** Offset (expressed in bytes) for the source byte in the extracted fields vector.
RESERVED	7:6	00b	RSV	N/A	Reserved.
WSB_SRC_OFF	12:8	0x0	RW	UNDEFINED	**Source Word Offset** Offset (expressed in words) for the source word in the extracted fields vector.
RESERVED	31:13	0x0	RSV	N/A	Reserved.

13.2.2.13.4 DWord Selection Base Select per Profile - GL_ACL_PROFILE_DWSB_SEL[n] (0x00391088 + 0x4*n, n=0...15; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DWORD_SEL_OFF	3:0	0x0	RW	UNDEFINED	**DWord Selection Offset** Offset (expressed in DWords) for the source DWord in the extracted fields vector
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.13.5 Profile Assignment to Scenario - GL_ACL_PROFILE_PF_CFG[n] (0x003910C8 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SCEN_SEL	5:0	0x0	RW	UNDEFINED	**Scenario Selection** This determines the selected scenario for this PF when this profile is selected.
RESERVED	31:6	0x0	RSV	N/A	Reserved.

13.2.2.13.6 Range Checker Configuration per Profile - GL_ACL_PROFILE_RC_CFG[n] (0x003910E8 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LOW_BOUND	15:0	0x0	RW	UNDEFINED	**Low Boundary** The range checker's output is negative if the related field's value is lower than this value.
HIGH_BOUND	31:16	0x0	RW	UNDEFINED	**High Boundary** The range checker's output is negative if the related field's value is higher than this value.

13.2.2.13.7 Word Selection Base Range Checked Fields Masking per Profile - GL_ACL_PROFILE_RCF_MASK[n] (0x00391108 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MASK	15:0	0x0	RW	UNDEFINED	**Mask** Range checker input mask. When a bit is cleared in this field, the relevant bit is cleared in the range checker's input.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.13.8 Default Action Array - GL_ACL_DEFAULT_ACT[n] (0x00391168 + 0x4*n, n=0...15; RO)

Field definitions are the same as those defined in Section 13.2.2.13.15.

13.2.2.13.9 VSI Dependent ACL Configuration - VSI_ACL_DEF_SEL[VSI] (0x00391800 + 0x4*VSI, VSI=0...767; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RX_PROFILE_MISS_SEL	1:0	00b	RW	UNDEFINED	**Rx Profile Miss Selection** This field selects which action set sets the action when a packet belonging to this VSI is not associated with any profile (i.e., misses the profile table).
RESERVED	3:2	00b	RSV	N/A	Reserved.
RX_TABLES_MISS_SEL	5:4	00b	RW	UNDEFINED	**Rx Tables Miss Selection** This field selects which action set sets the action when a packet belonging to this VSI is not associated with any ACL action (i.e., misses the ACL tables).
RESERVED	7:6	00b	RSV	N/A	Reserved.
TX_PROFILE_MISS_SEL	9:8	00b	RW	UNDEFINED	**Tx Profile Miss Selection** This field selects which action set sets the action when an packet originating from this VSI is not associated with any profile (i.e., misses the profile table).
RESERVED	11:10	00b	RSV	N/A	Reserved.
TX_TABLES_MISS_SEL	13:12	00b	RW	UNDEFINED	**Tx Tables Miss Selection** This field selects which action set sets the action when a packet originating from this VSI is not associated with any ACL action (i.e., misses the ACL tables).

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	31:14	0x0	RSV	N/A	Reserved.

13.2.2.13.10 GL_ACL_CHICKEN_REGISTER - GL_ACL_CHICKEN_REGISTER (0x00393810; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCAM_DATA_POL_CH	0	0b	RW	UNDEFINED	**TCAM Data Polarity Chicken**
TCAM_ADDR_POL_CH	1	0b	RW	UNDEFINED	**TCAM Address Polarity Chicken**
RESERVED	31:2	0x0	RSV	N/A	Reserved.

13.2.2.13.11 TCAM Write Key Low - GL_ACL_TCAM_KEY_L (0x00393814; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GL_ACL_FFU_TCAM_KEY_L	31:0	0x0	RW	UNDEFINED	**TCAM Key Low** 32 LSB of the entry key to be written to the TCAM.

13.2.2.13.12 TCAM Write Key High - GL_ACL_TCAM_KEY_H (0x00393818; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GL_ACL_FFU_TCAM_KEY_H	7:0	0x0	RW	UNDEFINED	**TCAM Key High** 8 MSB of the entry key to be written to the TCAM.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.13.13 TCAM Write Key Invert Low - GL_ACL_TCAM_KEY_INV_L (0x0039381C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GL_ACL_FFU_TCAM_KEY_INV_L	31:0	0x0	RW	UNDEFINED	**TCAM Key Invert Low** 32 LSB of the key invert to be written to the TCAM.

13.2.2.13.14 TCAM Write Key Invert High - GL_ACL_TCAM_KEY_INV_H (0x00393820; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GL_ACL_FFU_TCAM_KEY_INV_H	7:0	0x0	RW	UNDEFINED	**TCAM Key Invert High** 8 MSB of the key invert to be written to the TCAM.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.13.15 Action Write Data - GL_ACL_ACTMEM_ACT[n] (0x00393824 + 0x4*n, n=0...1; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VALUE	15:0	0x0	RW	UNDEFINED	**Value** Action value.
RESERVED	19:16	0x0	RSV	N/A	Reserved.
MDID	25:20	0x0	RW	UNDEFINED	**MDID** Action metadata ID.
RESERVED	27:26	00b	RSV	N/A	Reserved.
PRIORITY	30:28	000b	RW	UNDEFINED	**Priority** Action priority.
RESERVED	31	0b	RSV	N/A	Reserved.

13.2.2.13.16 Scenario Configuration Write Data Low Part - GL_ACL_SCENARIO_CFG_L[n] (0x0039382C + 0x4*n, n=0...15; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SELECT0	6:0	0x0	RW	UNDEFINED	**Select 0** Selects byte 0 of the lookup key: • If *SELECT0* is in the range of 0-31, the value is: KEY16[SELECT0][7:0] • If *SELECT0* is in the range of 32-63, the value is: KEY8[SELECT0 - 32] • If *SELECT0* is in the range of 64-95, the value is 0x0. • If *SELECT0* is in the range of 96-111, the value is: KEY32[SELECT0 - 96][7:0]
RESERVED	7	0b	RSV	N/A	Reserved.
SELECT1	14:8	0x0	RW	UNDEFINED	**Select 1** Selects byte 1 of the lookup key: • If *SELECT1* is in the range of 0-31, the value is: KEY16[SELECT1][15:8] • If *SELECT1* is in the range of 32-63, the value is: KEY8[SELECT1 - 32]. • If *SELECT1* is in the range of 64-95, the value is 0x0. • If *SELECT1* is in the range of 96-111, the value is: KEY32[SELECT1 - 96][15:8]
RESERVED	15	0b	RSV	N/A	Reserved.
SELECT2	22:16	0x0	RW	UNDEFINED	**Select 2** Selects byte 2 of the lookup key: • If *SELECT2* is in the range of 0-31, the value is: KEY16[SELECT2][7:0] • If *SELECT2* is in the range of 32-63, the value is: KEY8[SELECT2 - 32]. • If *SELECT2* is in the range of 64-95, the value is 0x0. • If *SELECT2* is in the range of 96-111, the value is: KEY32[SELECT2 - 96][23:16]
RESERVED	23	0b	RSV	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
SELECT3	30:24	0x0	RW	UNDEFINED	**Select 3** Selects byte 3 of the lookup key: • If *SELECT3* is in the range of 0-31, the value is: KEY16[SELECT3][15:8] • If *SELECT3* is in the range of 32-63, the value is: KEY8[SELECT3 - 32]. • If *SELECT3* is in the range of 64-95, the value is 0x0. • If *SELECT3* is in the range of 96-111, the value is: KEY32[SELECT3 - 96][31:24]
RESERVED	31	0b	RSV	N/A	Reserved.

13.2.2.13.17 Scenario Configuration Write Data High Part - GL_ACL_SCENARIO_CFG_H[n] (0x0039386C + 0x4*n, n=0...15; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SELECT4	4:0	0x0	RW	UNDEFINED	**Select 4** Selects byte 4 of the lookup key: Selects KEY8[SelectTop] as the top eight bits of the lookup key. Only KEY8s can be selected for this field.
RESERVED	7:5	000b	RSV	N/A	Reserved.
CHUNKMASK	15:8	0x0	RW	UNDEFINED	**Chunk Mask** Specifies the validity of 64-rule chunks: • ChunkMask[0] enables rules 0..63. • ChunkMask[1] enables rules 64..127. • and so on Used to create smaller per-scenario tables.
RESERVED	23:16	0x0	RSV	N/A	Reserved.
START_COMPARE	24	0b	RW	UNDEFINED	**Start Compare** Starts a compare cascade.
RESERVED	27:25	000b	RSV	N/A	Reserved.
START_SET	28	0b	RW	UNDEFINED	**Start Set** Start a TCAM stacking.
RESERVED	31:29	000b	RSV	N/A	Reserved.

13.2.2.13.18 Scenario Action RAM Configuration Write Data - GL_ACL_SCENARIO_ACT_CFG[n] (0x003938AC + 0x4*n, n=0...19; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
ACTMEM_SEL	3:0	0x0	RW	UNDEFINED	**Action Memory Select** Selects which TCAM controls the action memory for this scenario
RESERVED	7:4	0x0	RSV	N/A	Reserved.
ACTMEM_EN	8	0b	RW	UNDEFINED	**Action Memory Enable** Enable/disable control: 0b = The action memory is disable for this scenario. 1b = The action memory is enabled for this scenario.
RESERVED	31:9	0x0	RSV	N/A	Reserved.

13.2.2.14 PF - Rx Filters Registers

13.2.2.14.1 Global Classification Filter - PE Table Clear - GLQF_PETABLE_CLR[n] (0x000AA078 + 0x4*n, n=0...1; RW)

Register '0' is used by the EMP as part of PFR / VFR flows.
Register '1' is used by the PE as part of a function table removal flow.

Field	Bit(s)	Init.	Type	CFG Policy	Description
VM_VF_NUM	9:0	0x0	RW	UNDEFINED	**VM/VF Number** Absolute VF number. Relevant only for *VM_VF_TYPE* = VF (00b). Otherwise, must be set to zero.
VM_VF_TYPE	11:10	00b	RW	UNDEFINED	**VM/VF Type** Defines the impact of the clear action: 00b = VF clear. 10b = PF clear. All other values are reserved.
PF_NUM	14:12	000b	RW	UNDEFINED	**PF Number** PF index.
RESERVED	15	0b	RSV	N/A	Reserved.
PE_BUSY	16	0b	SC	UNDEFINED	**PE Busy** The *BUSY* flag indicates that PE cache clear is in progress. It is set by software and cleared by the device.
PE_CLEAR	17	0b	RW	UNDEFINED	**PE Clear** Setting this flag clears the PE filters entries of the entity (PF/VF) from the PE cache.
RESERVED	31:18	0x0	RSV	N/A	Reserved.

13.2.2.14.2 PF Classification Filter - PE Field Vector Bitmap Enable - GLQF_PE_FVE (0x0020E514; RW)

This register is used by the PE to program QH filter entries.

Field	Bit(s)	Init.	Type	CFG Policy	Description
W_ENA	23:0	0x0	RW	UNDEFINED	**Word Enable** Bit 'i' in this field enables word 'i' in the field vector.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.14.3 Global Classification Filter - Hash Input Set - GLQF_HINSET[n,m] (0x0040E000 + 0x4*n + 0x200*m, n=0...127, m=0...5; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FV_WORD_INDX0	4:0	0x1F	RW	UNDEFINED	**FV Word Index 0** The word index in the FV that is copied to the input set word “m*4” of packet profile “n”, where “n”, “m” are the register's indexes in the array.
RESERVED	6:5	00b	RSV	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FV_WORD_VAL0	7	1b	RW	UNDEFINED	**FV Word Valid 0** Valid indication for the word defined by *FV_WORD_INDX0*.
FV_WORD_INDX1	12:8	0x1F	RW	UNDEFINED	**FV Word Index 1** The word index in the FV that is copied to the input set word "m*4+1" of packet profile "n", where "n", "m" are the register's indexes in the array.
RESERVED	14:13	00b	RSV	N/A	Reserved.
FV_WORD_VAL1	15	1b	RW	UNDEFINED	**FV Word Valid 1** Valid indication for the word defined by *FV_WORD_INDX1*.
FV_WORD_INDX2	20:16	0x1F	RW	UNDEFINED	**FV Word Index 2** The word index in the FV that is copied to the input set word "m*4+2" of packet profile "n", where "n", "m" are the register's indexes in the array.
RESERVED	22:21	00b	RSV	N/A	Reserved.
FV_WORD_VAL2	23	1b	RW	UNDEFINED	**FV Word Valid 2** Valid indication for the word defined by *FV_WORD_INDX2*.
FV_WORD_INDX3	28:24	0x1F	RW	UNDEFINED	**FV Word Index 3** The word index in the FV that is copied to the input set word "m*4+3" of packet profile "n", where "n", "m" are the register's indexes in the array.
RESERVED	30:29	00b	RSV	N/A	Reserved.
FV_WORD_VAL3	31	1b	RW	UNDEFINED	**FV Word Valid 3** Valid indication for the word defined by *FV_WORD_INDX3*.

13.2.2.14.4 Global Classification Filter - Symmetric Hash - GLQF_HSYMM[n,m] (0x0040F000 + 0x4*n + 0x200*m, n=0...127, m=0...5; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FV_SYMM_INDX0	4:0	0x0	RW	UNDEFINED	**FV Symmetric Index 0** Registers "n" belongs to packet profile "n". There are multiple registers per packet profile indexed by "m". If the *SYMM0_ENA* is set, the input set word index "m*4" equals an XOR function between FV word index = *FV_SYMM_INDX0* (in this register) and FV word index = *FV_INSET_INDX0* (in the GLQF_HINSET register).
RESERVED	6:5	00b	RSV	N/A	Reserved.
SYMM0_ENA	7	0b	RW	UNDEFINED	**Symmetric 0 Enable** Symmetric word indication for the word defined by *FV_SYMM_INDX0*.
FV_SYMM_INDX1	12:8	0x0	RW	UNDEFINED	**FV Symmetric Index 1** Registers "n" belongs to packet profile "n". There are multiple registers per packet profile indexed by "m". If the *SYMM1_ENA* is set, the input set word index "m*4+1" equals an XOR function between FV word index = *FV_SYMM_INDX1* (in this register) and FV word index = *FV_INSET_INDX1* (in the GLQF_HINSET register).
RESERVED	14:13	00b	RSV	N/A	Reserved.
SYMM1_ENA	15	0b	RW	UNDEFINED	**Symmetric 1 Enable** Symmetric word indication for the word defined by *FV_SYMM_INDX1*.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FV_SYMM_INDX2	20:16	0x0	RW	UNDEFINED	**FV Symmetric Index 2** Registers "n" belongs to packet profile "n". There are multiple registers per packet profile indexed by "m". If the *SYMM2_ENA* is set, the input set word index "m*4+2" equals an XOR function between FV word index = *FV_SYMM_INDX2* (in this register) and FV word index = *FV_INSET_INDX2* (in the GLQF_HINSET register).
RESERVED	22:21	00b	RSV	N/A	Reserved.
SYMM2_ENA	23	0b	RW	UNDEFINED	**Symmetric 2 Enable** Symmetric word indication for the word defined by *FV_SYMM_INDX2*.
FV_SYMM_INDX3	28:24	0x0	RW	UNDEFINED	**FV Symmetric Index 3** Registers "n" belongs to packet profile "n". There are multiple registers per packet profile indexed by "m". If the *SYMM3_ENA* is set, the input set word index "m*4+3" equals an XOR function between FV word index = *FV_SYMM_INDX3* (in this register) and FV word index = *FV_INSET_INDX3* (in the GLQF_HINSET register).
RESERVED	30:29	00b	RSV	N/A	Reserved.
SYMM3_ENA	31	0b	RW	UNDEFINED	**Symmetric 3 Enable** Symmetric word indication for the word defined by *FV_SYMM_INDX3*.

13.2.2.14.5 Global Classification Filter - Hash Mask - GLQF_HMASK[n] (0x0040FC00 + 0x4*n, n=0...31; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MSK_INDEX	4:0	0x0	RW	UNDEFINED	**Mask Index** Word index in the Input Set vector to be masked.
RESERVED	15:5	0x0	RSV	N/A	Reserved.
MASK	31:16	0x0	RW	UNDEFINED	**Mask** Mask word. Any bit set to "0" masks out the matched bit of *MSK_INDEX* byte in the field vector.

13.2.2.14.6 Global Classification Filter - Hash Mask Select - GLQF_HMASK_SEL[n] (0x00410000 + 0x4*n, n=0...127; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MASK_SEL	31:0	0x0	RW	UNDEFINED	**Mask Select** The *MASK_SEL* is a bitmap field that enables the GLQF_HMASK registers per packet profile "n", where "n" is the GLQF_HMASK_SEL register index. Bit "x" in this register enables register GLQF_HMASK[x].

13.2.2.14.7 Global Classification Filter - FD Mask Select - GLQF_FDMASK_SEL[n] (0x00410400 + 0x4*n, n=0...127; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MASK_SEL	31:0	0x0	RW	UNDEFINED	**Mask Select** The *MASK_SEL* is a bitmap field that enables the GLQF_FDMASK registers per packet profile "n", where "n" is the GLQF_FDMASK_SEL register index. Bit "x" in this register enables register GLQF_FDMASK[x].

13.2.2.14.8 Global Classification Filter - FD Mask - GLQF_FDMASK[n] (0x00410800 + 0x4*n, n=0...31; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MSK_INDEX	4:0	0x0	RW	UNDEFINED	**Mask Index** Word index in the Input Set vector to be masked.
RESERVED	15:5	0x0	RSV	N/A	Reserved.
MASK	31:16	0x0	RW	UNDEFINED	**Mask** Mask word. Any bit set to "0" masks out the matched bit of *MSK_INDEX* byte in the field vector.

13.2.2.14.9 Global Classification Filter - FD Input Set - GLQF_FDINSET[n,m] (0x00412000 + 0x4*n + 0x200*m, n=0...127, m=0...5; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FV_WORD_INDX0	4:0	0x1F	RW	UNDEFINED	**FV Word Index 0** The word index in the FV that is copied to the input set word "m*4" of packet profile "n", where "n", "m" are the register's indexes in the array.
RESERVED	6:5	00b	RSV	N/A	Reserved.
FV_WORD_VAL0	7	1b	RW	UNDEFINED	**FV Word Valid 0** Valid indication for the word defined by *FV_WORD_INDX0*.
FV_WORD_INDX1	12:8	0x1F	RW	UNDEFINED	**FV Word Index 1** The word index in the FV that is copied to the input set word "m*4+1" of packet profile "n", where "n", "m" are the register's indexes in the array.
RESERVED	14:13	00b	RSV	N/A	Reserved.
FV_WORD_VAL1	15	1b	RW	UNDEFINED	**FV Word Valid 1** Valid indication for the word defined by *FV_WORD_INDX1*.
FV_WORD_INDX2	20:16	0x1F	RW	UNDEFINED	**FV Word Index 2** The word index in the FV that is copied to the input set word "m*4+2" of packet profile "n", where "n", "m" are the register's indexes in the array.
RESERVED	22:21	00b	RSV	N/A	Reserved.
FV_WORD_VAL2	23	1b	RW	UNDEFINED	**FV Word Valid 2** Valid indication for the word defined by *FV_WORD_INDX2*.
FV_WORD_INDX3	28:24	0x1F	RW	UNDEFINED	**FV Word Index 3** The word index in the FV that is copied to the input set word "m*4+3" of packet profile "n", where "n", "m" are the register's indexes in the array.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	30:29	00b	RSV	N/A	Reserved.
FV_WORD_VAL3	31	1b	RW	UNDEFINED	**FV Word Valid 3** Valid indication for the word defined by *FV_WORD_INDX3*.

13.2.2.14.10 Global Classification Filter - FD SWAP - GLQF_FDSWAP[n,m] (0x00413000 + 0x4*n + 0x200*m, n=0...127, m=0...5; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FV_WORD_INDX0	4:0	0x0	RW	UNDEFINED	**FV Word Index 0** The word index in the FV that is copied to the swapped input set word “m*4” of packet profile “n”, where “n”, “m” are the register's indexes in the array.
RESERVED	6:5	00b	RSV	N/A	Reserved.
FV_WORD_VAL0	7	0b	RW	UNDEFINED	**FV Word Valid 0** Valid indication for the word defined by *FV_WORD_INDX0*.
FV_WORD_INDX1	12:8	0x0	RW	UNDEFINED	**FV Word Index 1** The word index in the FV that is copied to the swapped input set word “m*4+1” of packet profile “n”, where “n”, “m” are the register's indexes in the array.
RESERVED	14:13	00b	RSV	N/A	Reserved.
FV_WORD_VAL1	15	0b	RW	UNDEFINED	**FV Word Valid 1** Valid indication for the word defined by *FV_WORD_INDX1*.
FV_WORD_INDX2	20:16	0x0	RW	UNDEFINED	**FV Word Index 2** The word index in the FV that is copied to the swapped input set word “m*4+2” of packet profile “n”, where “n”, “m” are the register's indexes in the array.
RESERVED	22:21	00b	RSV	N/A	Reserved.
FV_WORD_VAL2	23	0b	RW	UNDEFINED	**FV Word Valid 2** Valid indication for the word defined by *FV_WORD_INDX2*.
FV_WORD_INDX3	28:24	0x0	RW	UNDEFINED	**FV Word Index 3** The word index in the FV that is copied to the swapped input set word “m*4+3” of packet profile “n”, where “n”, “m” are the register's indexes in the array.
RESERVED	30:29	00b	RSV	N/A	Reserved.
FV_WORD_VAL3	31	0b	RW	UNDEFINED	**FV Word Valid 3** Valid indication for the word defined by *FV_WORD_INDX3*.

13.2.2.14.11 Global Classification Filter - PE Input Set - GLQF_PEINSET[n,m] (0x00415000 + 0x4*n + 0x80*m, n=0...31, m=0...5; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FV_WORD_INDX0	4:0	0x1F	RW	UNDEFINED	**FV Word Index 0** The word index in the FV that is copied to the input set word “m*4” of packet profile “n”, where “n”, “m” are the register's indexes in the array.
RESERVED	6:5	00b	RSV	N/A	Reserved.
FV_WORD_VAL0	7	1b	RW	UNDEFINED	**FV Word Valid 0** Valid indication for the word defined by *FV_WORD_INDX0*.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FV_WORD_INDX1	12:8	0x1F	RW	UNDEFINED	**FV Word Index 1** The word index in the FV that is copied to the input set word "m*4+1" of packet profile "n", where "n", "m" are the register's indexes in the array.
RESERVED	14:13	00b	RSV	N/A	Reserved.
FV_WORD_VAL1	15	1b	RW	UNDEFINED	**FV Word Valid 1** Valid indication for the word defined by *FV_WORD_INDX1*.
FV_WORD_INDX2	20:16	0x1F	RW	UNDEFINED	**FV Word Index 2** The word index in the FV that is copied to the input set word "m*4+2" of packet profile "n", where "n", "m" are the register's indexes in the array.
RESERVED	22:21	00b	RSV	N/A	Reserved.
FV_WORD_VAL2	23	1b	RW	UNDEFINED	**FV Word Valid 2** Valid indication for the word defined by *FV_WORD_INDX2*.
FV_WORD_INDX3	28:24	0x1F	RW	UNDEFINED	**FV Word Index 3** The word index in the FV that is copied to the input set word "m*4+3" of packet profile "n", where "n", "m" are the register's indexes in the array.
RESERVED	30:29	00b	RSV	N/A	Reserved.
FV_WORD_VAL3	31	1b	RW	UNDEFINED	**FV Word Valid 3** Valid indication for the word defined by *FV_WORD_INDX3*.

13.2.2.14.12 Global Classification Filter - PE Mask - GLQF_PEMASK[n] (0x00415400 + 0x4*n, n=0...15; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MSK_INDEX	4:0	0x0	RW	UNDEFINED	**Mask Index** Word index in the Input Set vector to be masked.
RESERVED	15:5	0x0	RSV	N/A	Reserved.
MASK	31:16	0x0	RW	UNDEFINED	**Mask** Mask word. Any bit set to "0" masks out the matched bit of *MSK_INDEX* byte in the field vector.

13.2.2.14.13 Global Classification Filter - PE Mask Select - GLQF_PEMASK_SEL[n] (0x00415500 + 0x4*n, n=0...31; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MASK_SEL	15:0	0x0	RW	UNDEFINED	**Mask Select** The *MASK_SEL* is a bitmap field that enables the GLQF_PEMASK registers per packet profile "n", where "n'" is the GLQF_PEMASK_SEL register index. Bit "x" in this register enables register GLQF_PEMASK[x].
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.14.14 PF Classification Filter - Hash LUT - PFQF_HLUT[n,PF] (0x00430000 + 0x40*n + 0x4*PF, n=0...511, PF=0...7; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LUT0	7:0	0x0	RW	UNDEFINED	**LUT 0** Hash redirection LUT entry 4 x "n", where "n" is the register index.
LUT1	15:8	0x0	RW	UNDEFINED	**LUT 1** Hash redirection LUT entry 4 x "n" + 1, where "n" is the register index.
LUT2	23:16	0x0	RW	UNDEFINED	**LUT 2** Hash redirection LUT entry 4 x "n" + 2, where "n" is the register index.
LUT3	31:24	0x0	RW	UNDEFINED	**LUT 3** Hash redirection LUT entry 4 x "n" + 3, where "n" is the register index.

13.2.2.14.15 Global Classification Filter - Hash LUT - GLQF_HLUT[n,m] (0x00438000 + 0x4*n + 0x200*m, n=0...127, m=0...15; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LUT0	5:0	0x0	RW	UNDEFINED	**LUT 0** Hash redirection LUT entry 4 x "n", where "n" is the register index.
RESERVED	7:6	00b	RSV	N/A	Reserved.
LUT1	13:8	0x0	RW	UNDEFINED	**LUT 1** Hash redirection LUT entry 4 x "n" + 1, where "n" is the register index.
RESERVED	15:14	00b	RSV	N/A	Reserved.
LUT2	21:16	0x0	RW	UNDEFINED	**LUT 2** Hash redirection LUT entry 4 x "n" + 2, where "n" is the register index.
RESERVED	23:22	00b	RSV	N/A	Reserved.
LUT3	29:24	0x0	RW	UNDEFINED	**LUT 3** Hash redirection LUT entry 4 x "n" + 3, where "n" is the register index.
RESERVED	31:30	00b	RSV	N/A	Reserved.

13.2.2.14.16 PF Classification Filter - FD Enable - PFQF_FD_ENA (0x0043A000; RW)

Setting this register, the hardware auto-clears the internal PE Quad Hash context counter and table pointers of the function.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FD_ENA	0	0b	RW	UNDEFINED	**FD Enable** Enables programming and filtering of the FD filter for the function.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.14.17 PF Classification Filter - PE Enable - PFQF_PE_FILTERING_ENA (0x0043A080; RW)

Setting this register, the hardware auto-clears the internal PE Quad Hash context counter and table pointers of the function.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PE_ENA	0	0b	RW	UNDEFINED	**PE Enable** Enables the filtering and programming of PE filters.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.14.18 Global Classification Filter - Packet Profile to Hash TC Region Mapping - GLQF_PROF2TC[n,m] (0x0044D000 + 0x4*n + 0x200*m, n=0...127, m=0...3; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
OVERRIDE_ENA_0	0	0b	RW	UNDEFINED	**Override Enable 0** Enable for Override TC Region "8*m" in the Prof2TC table, for packet profile "n", where "n" is the register index and "m" is the word index. The index "8*m" in the table is selected by the *TC_OVER_SEL* field in the VSIQF_HASH_CTL register.
REGION_0	3:1	000b	RW	UNDEFINED	**Region 0** Receive queue region for Override TC Region "8*m" in the Prof2TC table, for packet profile "n", where 'n' is the register index and "m" is the word index. The index "8*m" in the table is selected by the *TC_OVER_SEL* field in the VSIQF_HASH_CTL register. This field is meaningful only if the *OVERRIDE_ENA_0* flag is set.
OVERRIDE_ENA_1	4	0b	RW	UNDEFINED	**Override Enable 1** Enable for Override TC Region "8*m+1" in the Prof2TC table, for packet profile "n", where "n" is the register index and "m" is the word index. The index "8*m+1" in the table is selected by the *TC_OVER_SEL* field in the VSIQF_HASH_CTL register.
REGION_1	7:5	000b	RW	UNDEFINED	**Region 1** Receive queue region for Override TC Region "8*m+1" in the Prof2TC table, for packet profile "n", where 'n' is the register index and "m" is the word index. The index "8*m+1" in the table is selected by the *TC_OVER_SEL* field in the VSIQF_HASH_CTL register. This field is meaningful only if the *OVERRIDE_ENA_1* flag is set.
OVERRIDE_ENA_2	8	0b	RW	UNDEFINED	**Override Enable 2** Enable for Override TC Region "8*m+2" in the Prof2TC table, for packet profile "n", where "n" is the register index and "m" is the word index. The index "8*m+2" in the table is selected by the *TC_OVER_SEL* field in the VSIQF_HASH_CTL register.
REGION_2	11:9	000b	RW	UNDEFINED	**Region 2** Receive queue region for Override TC Region "8*m+2" in the Prof2TC table, for packet profile "n", where 'n' is the register index and "m" is the word index. The index "8*m+2" in the table is selected by the *TC_OVER_SEL* field in the VSIQF_HASH_CTL register. This field is meaningful only if the *OVERRIDE_ENA_2* flag is set.

Field	Bit(s)	Init.	Type	CFG Policy	Description
OVERRIDE_ENA_3	12	0b	RW	UNDEFINED	**Override Enable 3** Enable for Override TC Region "8*m+3" in the Prof2TC table, for packet profile "n", where "n" is the register index and "m" is the word index. The index "8*m+3" in the table is selected by the *TC_OVER_SEL* field in the VSIQF_HASH_CTL register.
REGION_3	15:13	000b	RW	UNDEFINED	**Region 3** Receive queue region for Override TC Region "8*m+3" in the Prof2TC table, for packet profile "n", where 'n' is the register index and "m" is the word index. The index "8*m+3" in the table is selected by the *TC_OVER_SEL* field in the VSIQF_HASH_CTL register. This field is meaningful only if the *OVERRIDE_ENA_3* flag is set.
OVERRIDE_ENA_4	16	0b	RW	UNDEFINED	**Override Enable 4** Enable for Override TC Region "8*m+4" in the Prof2TC table, for packet profile "n", where "n" is the register index and "m" is the word index. The index "8*m+4" in the table is selected by the *TC_OVER_SEL* field in the VSIQF_HASH_CTL register.
REGION_4	19:17	000b	RW	UNDEFINED	**Region 4** Receive queue region for Override TC Region "8*m+4" in the Prof2TC table, for packet profile "n", where 'n' is the register index and "m" is the word index. The index "8*m+4" in the table is selected by the *TC_OVER_SEL* field in the VSIQF_HASH_CTL register. This field is meaningful only if the *OVERRIDE_ENA_4* flag is set.
OVERRIDE_ENA_5	20	0b	RW	UNDEFINED	**Override Enable 5** Enable for Override TC Region "8*m+5" in the Prof2TC table, for packet profile "n", where "n" is the register index and "m" is the word index. The index "8*m+5" in the table is selected by the *TC_OVER_SEL* field in the VSIQF_HASH_CTL register.
REGION_5	23:21	000b	RW	UNDEFINED	**Region 5** Receive queue region for Override TC Region "8*m+5" in the Prof2TC table, for packet profile "n", where 'n' is the register index and "m" is the word index. The index "8*m+5" in the table is selected by the *TC_OVER_SEL* field in the VSIQF_HASH_CTL register. This field is meaningful only if the *OVERRIDE_ENA_5* flag is set.
OVERRIDE_ENA_6	24	0b	RW	UNDEFINED	**Override Enable 6** Enable for Override TC Region "8*m+6" in the Prof2TC table, for packet profile "n", where "n" is the register index and "m" is the word index. The index "8*m+6" in the table is selected by the *TC_OVER_SEL* field in the VSIQF_HASH_CTL register.
REGION_6	27:25	000b	RW	UNDEFINED	**Region 6** Receive queue region for Override TC Region "8*m+6" in the Prof2TC table, for packet profile "n", where 'n' is the register index and "m" is the word index. The index "8*m+6" in the table is selected by the *TC_OVER_SEL* field in the VSIQF_HASH_CTL register. This field is meaningful only if the *OVERRIDE_ENA_6* flag is set.
OVERRIDE_ENA_7	28	0b	RW	UNDEFINED	**Override Enable 7** Enable for Override TC Region "8*m+7" in the Prof2TC table, for packet profile "n", where "n" is the register index and "m" is the word index. The index "8*m+7" in the table is selected by the *TC_OVER_SEL* field in the VSIQF_HASH_CTL register.

Field	Bit(s)	Init.	Type	CFG Policy	Description
REGION_7	31:29	000b	RW	UNDEFINED	**Region 7** Receive queue region for Override TC Region "8*m+7" in the Prof2TC table, for packet profile "n", where 'n' is the register index and "m" is the word index. The index "8*m+7" in the table is selected by the *TC_OVER_SEL* field in the VSIQF_HASH_CTL register. This field is meaningful only if the *OVERRIDE_ENA_7* flag is set.

13.2.2.14.19 Global Classification Filter Accelerated Port Bit Vector - GLQF_APBVT[n] (0x00450000 + 0x4*n, n=0...2047; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
APBVT	31:0	0x0	RW	UNDEFINED	**Accelerated Port Bit Vector** Each bit "i" in the APBVT of register "n" enables port number 32 x "n" + "i".

13.2.2.14.20 Global Classification Filter - FD Profile Evict Enable - GLQF_FDEVICTENA[n] (0x00452000 + 0x4*n, n=0...3; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FDEVICTENA	31:0	0x0	RW	UNDEFINED	**FD Evict Enable** Defines the PCTYPES that are candidates for hardware eviction of Flow Director filters. For register "n", any bit "k" in relation to profile [32*n+k], defines if a packet matching this profile is a candidate for auto-eviction by FIN/RST.

13.2.2.14.21 PF Classification Filter - QH TC Enable - PFQF_PE_TC_CTL (0x00452080; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TC_EN_PF	7:0	0x0	RW	UNDEFINED	**TC Enable PF** Bit "n" in this field enables TC "n" for PE filter for the PF.
RESERVED	15:8	0x0	RSV	N/A	Reserved.
TC_EN_VF	23:16	0x0	RW	UNDEFINED	**TC Enable VF** Bit "n" in this field enables TC "n" for PE filter for the VFs of the PF.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.14.22 Global Classification Filter - PE Control 2 - GLQF_PE_CTL2[n] (0x00455200 + 0x4*n, n=0...31; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TO_QH	1:0	00b	RW	UNDEFINED	**TO QH** The *TO_QH* field defines the target of the packet profile "n", where "n" is the register index. 00b = Reserved. 01b = Packet is candidate for the QH filter. 10b = Packet is candidate for the PE bypassing the QH filter. 11b = Packet is candidate for the QH filter and forwarded to the PE even if it does not match the filter.
APBVT_ENA	2	0b	RW	UNDEFINED	**APBVT Enable** Packet profile "n" should hit the APBVT to be a candidate for the QH or directly to the PE, where "n" is the register index.
RESERVED	31:3	0x0	RSV	N/A	Reserved.

13.2.2.14.23 Global Classification Filter - Hash LUT Size - GLQF_HLUT_SIZE[n] (0x00455400 + 0x4*n, n=0...15; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
HSIZE	0	0b	RW	UNDEFINED	**Hash Size** Defines the size of the matched Global Hash LUT. 0b = 128 1b = 512
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.14.24 PF Classification Filter - Hash LUT Size - PFQF_HLUT_SIZE (0x00455480; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
HSIZE	1:0	00b	RW	UNDEFINED	**Hash Size** This field defines the size of the PF Hash LUT. 00b = 128 01b = 512 10b = 2K 11b = Reserved
RESERVED	31:2	0x0	RSV	N/A	Reserved.

13.2.2.14.25 Global PE APBVT LAN Packet Counter - GLQF_PE_APBVT_CNT (0x00455500; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
APBVT_LAN	31:0	0x0	RCW	UNDEFINED	**APBVT LAN** Global counter of packets that matched the APBVT filter that are not directed to the PE just because they miss the QH filter. This counter is expected to provide an indication for the APBVT filter efficiency. ***Note:*** Count is not enabled if the packet is not initiating filter lookup (that is, not enabled by PF/VSI. Profile or packet is RoCE_UC).

13.2.2.14.26 VF Classification Filter - PE Enable - VPQF_PE_FILTERING_ENA[VF] (0x00455800 + 0x4*VF, VF=0...255; RW)

Setting this register, the hardware auto-clears the internal PE Quad Hash context counter and table pointers of the function.

Field definitions are the same as those defined in Section 13.2.2.14.17.

13.2.2.14.27 Global Classification Filter - Hash Key - GLQF_HKEY[n] (0x00456000 + 0x4*n, n=0...12; RW)

Field definitions are the same as those defined in Section 13.2.2.12.18.

13.2.2.14.28 Global Classification Filter Control - GLQF_FD_CTL (0x00460000; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FDLONG	3:0	0x1	RW	N/A	**FD Long** FD filter entry found at larger location in the bucket than FDLONGB is reported as such in the receive descriptor. Value of zero, means all collision are reported. In addition, PFQF_FD_CLSN1 statistical counter is incremented for the relevant PF. See PFQF_FD_ST_CTL for details of statistical counter configuration.
HASH_REPORT	4	0b	RW	UNDEFINED	**Hash Report** When this flag is set, the FD Bucket_HASH and Bucket_LEN parameters are reported in the Filter Programming Status Descriptor.
FLT_ADDR_REPORT	5	0b	RW	UNDEFINED	**Filter Address Report** When this flag is set, FD actual FLT_Addr (on-die memory) parameters are reported in the Filter Programming Status Descriptor.
RESERVED	31:6	0x0	RSV	N/A	Reserved.

13.2.2.14.29 Global Classification Filter - FD Space Size - GLQF_FD_SIZE (0x00460010; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FD_GSIZE	14:0	0x0	RW	UNDEFINED	**Guaranteed Size** Global guaranteed number of filters in the FD table.
RESERVED	15	0b	RSV	N/A	Reserved.
FD_BSIZE	30:16	0x0	RW	UNDEFINED	**Best Size** Global best effort number of filters in the FD table,
RESERVED	31	0b	RSV	N/A	Reserved.

13.2.2.14.30 Global Classification Filter - FD Space Counters - GLQF_FD_CNT (0x00460018; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FD_GCNT	14:0	0x0	RW/V	UNDEFINED	**Guaranteed Count** Global guaranteed filter counter in the FD table.
RESERVED	15	0b	RSV	N/A	Reserved.
FD_BCNT	30:16	0x0	RW/V	UNDEFINED	**Best Count** Global best effort filter counter in the FD table.
RESERVED	31	0b	RSV	N/A	Reserved.

13.2.2.14.31 PF Classification Filter - FD Space Sizes - PFQF_FD_SIZE (0x00460100; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FD_GSIZE	14:0	0x0	RW	UNDEFINED	**Guaranteed Size** PF guaranteed number of filters in the FD table.
RESERVED	15	0b	RSV	N/A	Reserved.
FD_BSIZE	30:16	0x0	RW	UNDEFINED	**Best Size** PF best effort number of filters in the FD table.
RESERVED	31	0b	RSV	N/A	Reserved.

13.2.2.14.32 Global Classification Filter - FD PF Space Counter - PFQF_FD_CNT (0x00460180; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FD_GCNT	14:0	0x0	RW/V	UNDEFINED	**Guaranteed Count** PF guaranteed filter counter in the FD table.
RESERVED	15	0b	RSV	N/A	Reserved.
FD_BCNT	30:16	0x0	RW/V	UNDEFINED	**Best Count** PF best effort filter counter in the FD table.
RESERVED	31	0b	RSV	N/A	Reserved.

13.2.2.14.33 Global Classification Filter - FD PF Space Counter - PFQF_FD_SUBTRACT (0x00460200; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FD_GCNT	14:0	0x0	WO	UNDEFINED	**Guaranteed Count** Subtract value for PF guaranteed filter counter in the FD table. Value written to this field will subtracted from GLQF_FD_CNT.*FD_GCNT* and PFQF_FD_CNT.*FD_GCNT* (for the same PF number). Reading this register returns 0.
RESERVED	15	0b	RSV	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FD_BCNT	30:16	0x0	WO	UNDEFINED	**Best Count** Subtract value for PF best effort filter counter in the FD table. Value written to this field will subtracted from GLQF_FD_CNT.*BFD_CNT* and PFQF_FD_CNT.*FD_BCNT* (for the same PF number). Reading this register will return 0.
RESERVED	31	0b	RSV	N/A	Reserved.

13.2.2.14.34 PF Classification Filter - PE Control - PFQF_PE_CTL1 (0x00470000; RW)

Setting this register, the hardware auto-clears the internal PE Quad Hash context counter and table pointers of the function.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PEHSIZE	3:0	0x0	RW	UNDEFINED	**Size** *PEHSIZE* defines the number of "buckets" of the PE Quad Hash filter table for the function defined in power of 2 equals 1K x 2 ** *PEHSIZE*. *PEHSIZE* can have any value between 0 and 10. Other values are reserved. *PEHSIZE* = 0, 1,... 10 is equivalent to 1K, 2K, 4K... 1M buckets.
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.14.35 PF Classification Filter - PE Control - PFQF_PE_CTL2 (0x00470040; RW)

Setting this register, the hardware auto-clears the internal PE Quad Hash context counter and table pointers of the function.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PEDSIZE	3:0	0x0	RW	UNDEFINED	**Size** *PEDSIZE* defines the number of PE Quad Hash contexts for the function defined in power of 2 equals to 0.5K x 2 ** *PEDSIZE*-1. *PEDSIZE* can have any value between 0 and 9. Other values are reserved. *PEDSIZE* = 0, 1,... 9 is equivalent to 0.5K-1 1K-1 2K-1, 4K-1... 256K-1 contexts.
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.14.36 PF Free List Head Array - PFQF_PE_FLHD (0x00470100; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FLHD	23:0	0x0	RW	UNDEFINED	**Free List Head** PF Filter Free List Head Pointer is initialized by software, but through operation is managed by hardware. It is a pointer to the Collision FPM, to the first available filter entry in the link list of all unused entries. Upon initialization (i.e. XLR, or function Clear command), software must write this register to zero. This is addition to cleanup of FPM collision area, which should be written to all zeros. After PF configuration is enabled, software must not write to this register. In case of PF initialization, the software is expected to clear and initialize all *FLHD* field of all the VFs of the PF. In case of ungraceful XLR, firmware must perform the register cleanup process.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.14.37 PF Classification Filter PE Filter Counter 0 - PFQF_PECNT_0 (0x00470200; RW)

Total bucket count.

Field	Bit(s)	Init.	Type	CFG Policy	Description
BUCKETCNT	17:0	0x0	RW/V	UNDEFINED	**Bucket Count** Reflects the number of active PE filter buckets of the function. This counter together with the PFQF_PECNT_1 register gives a hint on how uniform the filter's hash function is.
RESERVED	31:18	0x0	RSV	N/A	Reserved.

13.2.2.14.38 PF Classification Filter PE Filter Counter 1 - PFQF_PECNT_1 (0x00470300; RW)

Total filter count.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FLTCNT	17:0	0x0	RW/V	DYNAMIC	**Filter Count** Reflects the total number of active PE filters of the function.
RESERVED	31:18	0x0	RSV	N/A	Reserved.

13.2.2.14.39 PF Control Register for the Statistic Counter - PFQF_PE_ST_CTL (0x00470400; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PF_CNT_EN	0	0b	RW	UNDEFINED	**PE Count Enable** Enable counting PF filters. CLSN0 counts entry hits that are less than or equal to PELONGB. CLSN1 counts entry hits that are higher collision order than PELONGB. ***Note:*** Clearing this bit, does not clear the counter.

Field	Bit(s)	Init.	Type	CFG Policy	Description
VFS_CNT_EN	1	0b	RW	UNDEFINED	**VFs Count Enable** Enable counting filters of the PF and all VFs of this PF. CLSN0 counts entry hits that are less than or equal to PELONGB. CLSN1 counts entry hits that are higher collision order than PELONGB. ***Note:*** Clearing this bit, does not clear the counter.
VF_CNT_EN	2	0b	RW	UNDEFINED	**VF Count Enable** Enable counting specific VF of this PF, as specified by *VF_NUM*. CLSN0 counts entry hits that are less than or equal to PELONGB. CLSN1 counts entry hits that are higher collision order than PELONGB. ***Note:*** Clearing this bit, does not clear the counter.
RESERVED	15:3	0x0	RSV	N/A	Reserved.
VF_NUM	23:16	0x0	RW	UNDEFINED	**VF Number** VF number for specific VF counting.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.14.40 PF PE Classification Filter Collision Counter 0 - PFQF_PE_CLSN0 (0x00470480; RCW)

Filter lookup search at low bucket counter.

Field	Bit(s)	Init.	Type	CFG Policy	Description
HITSBCNT	31:0	0x0	RWC	UNDEFINED	**Hots Short Count** Counts processed packets of PF/VFs as defined by *_CNT_EN* in PFQF_PE_ST_CTL register, in which the PE filter is at a position shorter or equal than PELONGB threshold in the GLQF_FD_CTL register.

13.2.2.14.41 PF PE Classification Filter Collision Counter 1 - PFQF_PE_CLSN1 (0x00470500; RCW)

Filter lookup search at high bucket counter.

Field	Bit(s)	Init.	Type	CFG Policy	Description
HITLBCNT	31:0	0x0	RWC	UNDEFINED	**Hits Long Count** Counts processed packets of PF/VFs as defined by *_CNT_EN* in PFQF_PE_ST_CTL register, in which the PE filter is at a position longer than PELONGB threshold in the GLQF_FD_CTL register.

13.2.2.14.42 Global PE Classification Filter Outstanding Request Counter - GLQF_PE_OSR_STS (0x00471040; RCW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
QH_SRCH_MAXOSR	9:0	0x0	RCW	UNDEFINED	**QH Search Max Outstanding Search Requests** Reflects actual max number of active outstanding search requests (OSRs)
RESERVED	15:10	0x0	RSV	N/A	Reserved.
QH_CMD_MAXOSR	25:16	0x0	RCW	UNDEFINED	**QH Command Max Outstanding Search Requests** Reflects actual max number of active outstanding Add/Remove requests (OSRs)
RESERVED	31:26	0x0	RSV	N/A	Reserved.

13.2.2.14.43 QH ADD/REM Commands Status - GLQF_PE_CMD (0x00471080; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
ADDREM_STS	23:0	0x0	RW/V	UNDEFINED	**Add/Remove Statuses** 12x2-bit vector. Each 2-bit entry corresponds to an incoming command. Codes are: 00b = Free entry 01b = Busy (WIP) 10b = Done Fail 11b = Done Pass The statuses are written cyclically, enabling the monitoring of 12 commands in parallel. Busy indicates for software that the command is in processing and the next command can be inserted. Entries that are Done, are cleared automatically upon read. ***Note:*** Completions (Done) are not in order.
RESERVED	27:24	0x0	RSV	N/A	Reserved.
ADDREM_ID	31:28	0x0	RW/V	UNDEFINED	**Add/Remove Index** Reflects the index of the next free command location in the *ADDREM_STS* vector. The index is for the status entry, which is two bits. Therefore, for offset within the CSR, must to multiply by two.

13.2.2.14.44 Global Classification Filter Control - GLQF_PE_CTL (0x004710C0; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PELONG	3:0	0x1	RW	UNDEFINED	**PE Long** PE filter entry found at larger location in the bucket than PELONGB is reported as such in the receive descriptor. Value of one, means all collisions are reported. In addition, PFQF_PE_CLSN1 statistical counter is incremented for the relevant PF. See PFQF_PE_ST_CTL for details of statistical counter configuration.
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.14.45 VF Free List Head Array - VPQF_PE_FLHD[VF] (0x00472000 + 0x4*VF, VF=0...255; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FLHD	23:0	0x0	RW	UNDEFINED	**Free List Head** VF Filter Free List Head Pointer is initialized by software, but through operation is managed by hardware. It is a pointer to the Collision FPM, to the first available filter entry in the link list of all unused entries. Upon initialization (i.e. XLR, or function Clear command), software must write this register to zero. This is addition to cleanup of FPM collision area, which should be written to all zeros. After VF configuration is enabled, software must not write to this register.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.14.46 VF Classification Filter PE Filter Counter 0 - VPQF_PECNT_0[VF] (0x00472800 + 0x4*VF, VF=0...255; RW)

Total bucket count.

Field	Bit(s)	Init.	Type	CFG Policy	Description
BUCKETCNT	17:0	0x0	RW/V	UNDEFINED	**Bucket Count** Reflects the number of active PE filter buckets of the function. This counter together with the VPQF_PECNT_0 register gives a hint on how uniform the filter's hash function is.
RESERVED	31:18	0x0	RSV	N/A	Reserved.

13.2.2.14.47 VF Classification Filter PE Filter Counter 1 - VPQF_PECNT_1[VF] (0x00473000 + 0x4*VF, VF=0...255; RW)

Total filter count.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FLTCNT	17:0	0x0	RW/V	UNDEFINED	**Filter Count** Reflects the total number of active PE filters of the function.
RESERVED	31:18	0x0	RSV	N/A	Reserved

13.2.2.14.48 VF Classification Filter - PE Control - VPQF_PE_CTL1[VF] (0x00474000 + 0x4*VF, VF=0...255; RW)

Setting this register, the hardware auto-clears the internal PE Quad Hash context counter and table pointers of the function.

Field definitions are the same as those defined in Section 13.2.2.14.34.

13.2.2.14.49 PF Classification Filter - PE Control - VPQF_PE_CTL2[VF] (0x00474800 + 0x4*VF, VF=0...255; RW)

Setting this register, the hardware auto-clears the internal PE Quad Hash context counter and table pointers of the function.

Field definitions are the same as those defined in Section 13.2.2.14.35.

13.2.2.15 PF - Interrupt Registers

13.2.2.15.1 PF General Purpose IO Interrupt Enablement - PFINT_GPIO_ENA (0x00088080; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GPIO0_ENA	0	0b	RW	UNDEFINED	**GPIO 0 Enable** Enable interrupt on GPIO 0.
GPIO1_ENA	1	0b	RW	UNDEFINED	**GPIO 1 Enable** Enable interrupt on GPIO 1.
GPIO2_ENA	2	0b	RW	UNDEFINED	**GPIO 2 Enable** Enable interrupt on GPIO 2.
GPIO3_ENA	3	0b	RW	UNDEFINED	**GPIO 3 Enable** Enable interrupt on GPIO 3.
GPIO4_ENA	4	0b	RW	UNDEFINED	**GPIO 4 Enable** Enable interrupt on GPIO 4.
GPIO5_ENA	5	0b	RW	UNDEFINED	**GPIO 5 Enable** Enable interrupt on GPIO 5.
GPIO6_ENA	6	0b	RW	UNDEFINED	**GPIO 6 Enable** Enable interrupt on GPIO 6.
RESERVED	31:7	0x0	RSV	N/A	Reserved.

13.2.2.15.2 EMP General Purpose IO Interrupt Enablement - EMPINT_GPIO_ENA (0x000880C0; RW)

Field definitions are the same as those defined in Section 13.2.2.15.1.

13.2.2.15.3 VF Vector Allocation - VPINT_ALLOC_PCI[VF] (0x0009D000 + 0x4*VF, VF=0...255; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FIRST	10:0	0x0	RW	UNDEFINED	**First** Index of the first interrupt vector of the VF in the internal physical space.
RESERVED	11	0b	RSV	N/A	Reserved
LAST	22:12	0x0	RW	UNDEFINED	**Last** Index of the last interrupt vector of the VF in the internal physical space.
RESERVED	30:23	0x0	RSV	N/A	Reserved
VALID	31	0b	RW	UNDEFINED	**Valid** Valid indication. When this bit is clear, the contents of this CSR are not valid.

13.2.2.15.4 PF Vector Allocation - PCI - PFINT_ALLOC_PCI (0x0009D800; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FIRST	10:0	0x0	RW	UNDEFINED	**First** Index of the first interrupt vector of the PF in the internal physical space.
RESERVED	11	0b	RSV	N/A	Reserved
LAST	22:12	0x0	RW	UNDEFINED	**Last** Index of the last interrupt vector of the PF and its VFs in the internal physical space.
RESERVED	30:23	0x0	RSV	N/A	Reserved
VALID	31	0b	RW	UNDEFINED	**Valid** Valid indication. When this bit is clear, the contents of this CSR are not valid.

13.2.2.15.5 Transmit Queue Interrupt Cause Control - QINT_TQCTL[DBQM] (0x00140000 + 0x4*DBQM, DBQM=0...16383; RW)

Field definitions are the same as those defined in Section 13.2.2.15.6.

13.2.2.15.6 Receive Queue Interrupt Cause Control - QINT_RQCTL[QRX] (0x00150000 + 0x4*QRX, QRX=0...2047; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MSIX_INDX	10:0	0x0	RW	UNDEFINED	**MSI-X Index** MSI-X vector index within the function space. The software should set the *MSIX_INDX* to values in the range of allocated interrupt vectors to the function.
ITR_INDX	12:11	00b	RW	UNDEFINED	**ITR Index** ITR Index of the interrupt cause. 00b = ITR0 01b = ITR1 10b = ITR2 11b = No ITR
RESERVED	29:13	0x0	RSV	N/A	Reserved.
CAUSE_ENA	30	0b	RW	UNDEFINED	**Cause Enable** Enable interrupt by this queue. When *CAUSE_ENA* is cleared, interrupts are not generated by the queue. The queue remains in the interrupt linked list and is processed at ITR expiration.
RESERVED	31	0b	RSV	N/A	Reserved.

13.2.2.15.7 Global Interrupt Throttling - GLINT_ITR[n,INT] (0x00154000 + 0x2000*n + 0x4*INT, n=0...2, INT=0...2047; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
INTERVAL	11:0	0x0	RW	UNDEFINED	**Interval** ITR "n" interval, where "n" is the register index = 0,1,2 for the three ITRs per interrupt. It is defined in 2 µs units enabling interval range from zero to 8160 µs (0xFF0). Setting the *INTERVAL* to zero enables immediate interrupt. This register can be programmed also by setting the *INTERVAL* field in the matched xxINT_DYN_CTLx register.
RESERVED	31:12	0x0	RSV	N/A	Reserved.

13.2.2.15.8 Global Interrupt Rate Limit - GLINT_RATE[INT] (0x0015A000 + 0x4*INT, INT=0...2047; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
INTERVAL	5:0	0x0	RW	UNDEFINED	**Interval** Time interval defined in 4 µs units between consecutive credit incremental. When the interrupt rate limit is enabled by the *INTRL_ENA* flag in this register, the *INTERVAL* must be greater than zero. And for accurate rate limit, the *INTERVAL* must be smaller than 0x3C (up to 236-µs).
INTRL_ENA	6	0b	RW	UNDEFINED	**Interrupt Rate Limit Enable** Enable Interrupt Rate Limit on this interrupt vector.
RESERVED	31:7	0x0	RSV	N/A	Reserved.

13.2.2.15.9 Global PE Completion Event Queue Interrupt Cause Control - GLINT_CEQCTL[INT] (0x0015C000 + 0x4*INT, INT=0...2047; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MSIX_INDX	10:0	0x0	RW	UNDEFINED	**MSI-X Index** MSI-X vector index within the function space. The software should set the *MSIX_INDX* to values in the range of allocated interrupt vectors to the function.
ITR_INDX	12:11	0x0	RW	UNDEFINED	**ITR Index** ITR Index of the interrupt cause. 00b = ITR0 01b = ITR1 10b = ITR2 11b = No ITR
RESERVED	29:13	00b	RSV	N/A	Reserved.
CAUSE_ENA	30	0b	RW	UNDEFINED	**Cause Enable** Enable interrupt by this queue. When *CAUSE_ENA* is cleared, interrupts are not generated by the queue. The queue remains in the interrupt linked list and is processed at ITR expiration.
RESERVED	31	0b	RSV	N/A	Reserved.

13.2.2.15.10 Global Interrupt Dynamic Control - GLINT_DYN_CTL[INT] (0x00160000 + 0x4*INT, INT=0...2047; RW)

In case of MSI or Legacy INTA mode of operation Interrupt zero is the only valid interrupt. The space of these registers is according to the number of total interrupt registers for the device. The PF can access only those first "N" interrupts according to the PFINT_ALLOC registers.

Field	Bit(s)	Init.	Type	CFG Policy	Description
INTENA	0	0b	RW	UNDEFINED	**Interrupt Enable** 0b = Interrupt is disabled. 1b = Interrupt is enabled. See auto-clear policy in the Section 9.1.1.1, "Interrupt Enable Procedure". This bit is meaningful only if the *INTENA_MSK* flag in this register is not set. ***Note:*** The *INTENA* and *WB_ON_ITR* flags are mutually exclusive.
CLEARPBA	1	0b	RW1C	UNDEFINED	**Clear PBA** Setting this bit, the matched PBA bit is cleared. This bit is auto-cleared by the hardware.
SWINT_TRIG	2	0b	RW1C	DYNAMIC	**Software Interrupt Trigger** When the bit is set, a software interrupt is triggered. This bit is auto-cleared by the hardware.
ITR_INDX	4:3	00b	RW1C	DYNAMIC	**ITR Index** This field defines the ITR Index to be updated as follows: 00b = ITR0 01b = ITR1 10b = ITR2 11b = No ITR Update This field is auto-cleared by the hardware.
INTERVAL	16:5	0x0	RW1C	DYNAMIC	**Interval** The interval for the ITR defined by the *ITR_INDX* in this register. This field is auto-cleared by the hardware.
RESERVED	23:17	0x0	RSV	N/A	Reserved.
SW_ITR_INDX_ENA	24	0b	RW1C	DYNAMIC	**Software ITR Index Enable** This flag enables the programming of the *SW_ITR_INDX* in this register. This flag is auto cleared by the hardware.
SW_ITR_INDX	26:25	00b	RW	UNDEFINED	**Software ITR Index** ITR Index of the software interrupt. 00b = ITR0 01b = ITR1 10b = ITR2 11b = No ITR When programming this field, the *SW_ITR_INDX_ENA* flag in this register should be set as well.
RESERVED	29:27	000b	RSV	N/A	Reserved.
WB_ON_ITR	30	0b	RW	UNDEFINED	**Write Back on ITR** When this bit is set, ITR expiration triggers write back of completed descriptors without an interrupt. ***Note:*** The *INTENA* and *WB_ON_ITR* flags are mutually exclusive.
INTENA_MSK	31	0b	RW1C	DYNAMIC	**Interrupt Enable Mask** When this bit is set, the *INTENA* setting does not impact the device setting. This bit is auto-cleared by the hardware.

13.2.2.15.11 Global Interrupt Vector 2 Function Allocation - GLINT_VECT2FUNC[INT] (0x00162000 + 0x4*INT, INT=0...2047; RW)

These registers map the interrupts to their functions.

Field	Bit(s)	Init.	Type	CFG Policy	Description
VF_NUM	7:0	0x0	RW	UNDEFINED	**VF Number** An absolute VF index in the range of 0 to 255. It is meaningful only if the *IS_PF* flag in this register is cleared. Otherwise, it should be set to zero.
RESERVED	11:8	0x0	RSV	N/A	Reserved.
PF_NUM	14:12	000b	RW	UNDEFINED	**PF Number** The PF index can be set to 0-7.
RESERVED	15	0b	RSV	N/A	Reserved.
IS_PF	16	0b	RW	UNDEFINED	**Is PF** 0b = The queue belongs to the VF. In this case the *VF_NUM* is valid as well. 1b = The interrupt belongs to the PF.
RESERVED	31:17	0x0	RSV	N/A	Reserved.

13.2.2.15.12 VF Mailbox Queue Mapping to Interrupt Control - VPINT_MBX_CTL[VSI] (0x0016A000 + 0x4*VSI, VSI=0...767; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MSIX_INDX	10:0	0x0	RW	UNDEFINED	**MSI-X Index** MSI-X vector index within the function space. The software should set the *MSIX_INDX* to values in the range of allocated interrupt vectors to the function.
ITR_INDX	12:11	00b	RW	UNDEFINED	**ITR Index** ITR Index of the interrupt cause. 00b = ITR0 01b = ITR1 10b = ITR2 11b = No ITR
RESERVED	29:13	0x0	RSV	N/A	Reserved.
CAUSE_ENA	30	0b	RW	UNDEFINED	**Cause Enable** Enable interrupt. When *CAUSE_ENA* is cleared, interrupts are not generated and relevant events are disregarded by the interrupt block. This control is not relevant for OICR causes for which each cause is controlled in *XXINT_PICR_ENA*. For OICR, this bit value is disregarded by the hardware.
RESERVED	31	0b	RSV	N/A	Reserved.

13.2.2.15.13 VF Mailbox Queue Mapping to Interrupt Control - VPINT_MBX_CPM_CTL[VP128] (0x0016B000 + 0x4*VP128, VP128=0...127; RW)

Field definitions are the same as those defined in Section 13.2.2.15.12.

13.2.2.15.14 VF HLP Mailbox Queue Mapping to Interrupt Control - VPINT_MBX_HLP_CTL[VP16] (0x0016B200 + 0x4*VP16, VP16=0...15; RW)

Field definitions are the same as those defined in Section 13.2.2.15.12.

13.2.2.15.15 VF PSM Mailbox Queue Mapping to Interrupt Control - VPINT_MBX_PSM_CTL[VP16] (0x0016B240 + 0x4*VP16, VP16=0...15; RW)

Field definitions are the same as those defined in Section 13.2.2.15.12.

13.2.2.15.16 PF Mailbox Queue Mapping to Interrupt Control - PFINT_MBX_CTL (0x0016B280; RW)

Field definitions are the same as those defined in Section 13.2.2.15.12.

13.2.2.15.17 PF0 CPM Mailbox Queue Mapping to Interrupt Control - PF0INT_MBX_CPM_CTL (0x0016B2C0; RW)

Field definitions are the same as those defined in Section 13.2.2.15.12.

13.2.2.15.18 PF0 HLP Mailbox Queue Mapping to Interrupt Control - PF0INT_MBX_HLP_CTL (0x0016B2C4; RW)

Field definitions are the same as those defined in Section 13.2.2.15.12.

13.2.2.15.19 PF0 PSM Mailbox Queue Mapping to Interrupt Control - PF0INT_MBX_PSM_CTL (0x0016B2C8; RW)

Field definitions are the same as those defined in Section 13.2.2.15.12.

13.2.2.15.20 PF0 CPM SB Queue Mapping to Interrupt Control - PF0INT_SB_CPM_CTL (0x0016B2CC; RW)

Field definitions are the same as those defined in Section 13.2.2.15.12.

13.2.2.15.21 VF CPM SB Queue Mapping to Interrupt Control - VPINT_SB_CPM_CTL[VP128] (0x0016B400 + 0x4*VP128, VP128=0...127; RW)

Field definitions are the same as those defined in Section 13.2.2.15.12.

13.2.2.15.22 PF SB Queue Mapping to Interrupt Control - PFINT_SB_CTL (0x0016B600; RW)

Field definitions are the same as those defined in Section 13.2.2.15.12.

13.2.2.15.23 PF0 HLPSB Queue Mapping to Interrupt Control - PF0INT_SB_HLP_CTL (0x0016B640; RW)

Field definitions are the same as those defined in Section 13.2.2.15.12.

13.2.2.15.24 VF PE Asynchronous Event Queue Interrupt Cause Control - VPINT_AEQCTL[VF] (0x0016B800 + 0x4*VF, VF=0...255; RW)

Field definitions are the same as those defined in Section 13.2.2.15.33.

13.2.2.15.25 PF Firmware Admin Queue Mapping to Interrupt Control - PFINT_FW_CTL (0x0016C800; RW)

Field definitions are the same as those defined in Section 13.2.2.15.12.

13.2.2.15.26 Global Tools Firmware Admin Queue Mapping to Interrupt Control - GLINT_FW_TOOL_CTL (0x0016C840; RW)

Field definitions are the same as those defined in Section 13.2.2.15.12.

13.2.2.15.27 PF0 HLP Firmware Admin Queue Mapping to Interrupt Control - PF0INT_FW_HLP_CTL (0x0016C844; RW)

Field definitions are the same as those defined in Section 13.2.2.15.12.

13.2.2.15.28 PF0 PSM Firmware Admin Queue Mapping to Interrupt Control - PF0INT_FW_PSM_CTL (0x0016C848; RW)

Field definitions are the same as those defined in Section 13.2.2.15.12.

13.2.2.15.29 PF Interrupt Other Cause Enablement - PFINT_OICR_ENA (0x0016C900; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	0	0b	RSV	N/A	Reserved.
INT_ENA	31:1	0x0	RW	UNDEFINED	**Interrupt Enable** Each bit set to '1' in this field enables its matched interrupt cause in the PFINT_OICR register.

13.2.2.15.30 Global Interrupt TimeSync PHY Mask - PFINT_TSYN_MSK (0x0016C980; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PHY_INDX	4:0	0x0	RW	UNDEFINED	**PHY Index** Bit 'i' in this register enables Quad-PHY index 'i' interrupt to the *TSYN_TX* flag in the PFINT_OICR register. This value should be loaded by NVM and correspond to the port-to-PF mapping.
RESERVED	31:5	0x0	RSV	N/A	Reserved

13.2.2.15.31 PF Interrupt Other Cause - PFINT_OICR (0x0016CA00; RCW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	0	1b	RSV	N/A	Reserved.
QUEUE	1	1b	RCW	UNDEFINED	**Queue** Interrupt indication for LAN queues and PE CEQs that are linked to the other cause interrupt vector.
RESERVED	9:2	0x0	RSV	N/A	Reserved.
HH_COMP	10	0b	RCW	UNDEFINED	**HH Complete** Hammock Harbor sequence is completed (matched completion message is received).
TSYN_TX	11	0b	RCW	UNDEFINED	**TimeSync Tx** Tx packet time is sampled.
TSYN_EVNT	12	0b	RCW	UNDEFINED	**TimeSync Event** Event is sampled by the 1588 timer due to a transition in one of the input GPIOs.
TSYN_TGT	13	0b	RCW	UNDEFINED	**TimeSync Target** One of the target time of the 1588 timer is expired.
HLP_RDY	14	0b	RCW	UNDEFINED	**HLP Ready** HLP changed its status from not RDY to RDY or visa versa.
CPM_RDY	15	0b	RCW	UNDEFINED	**CMP Ready** CPM changed its status from not RDY to RDY or visa versa.
ECC_ERR	16	0b	RCW	UNDEFINED	**ECC Error** Unrecoverable ECC Error. This bit is set when an unrecoverable error is detected in one of the device memories.
RESERVED	18:17	00b	RSV	N/A	Reserved.
MAL_DETECT	19	0b	RCW	UNDEFINED	**Malicious Detected** Malicious programming detected.
GRST	20	0b	RCW	UNDEFINED	**Global Resets Requested** CORER, GLOBR, or EMPR.
RESERVED	21	0b	RSV	N/A	Reserved.
GPIO	22	0b	RCW	UNDEFINED	**GPIO** GPIO Event indicates an event on any of the GPIO pins enabled for interrupt by the PFINT_GPIOCTL register. The GPIO state can be fetched on the GLGEN_GPIO_STAT register. The level transition that generates an interrupt is set for GPIO 'n' by the INT_MODE field in the matched GLGEN_GPIO_CTL[n] register.
RESERVED	23	0b	RSV	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
STORM_DETECT	24	0b	RCW	UNDEFINED	**Storm Detected** Indicates a change entering the storm control state of the LAN port that is connected to this PF. The storm control state is reflected in the PRT_SWT_SCSTS register.
RESERVED	25	0b	RSV	N/A	Reserved.
HMC_ERR	26	0b	RCW	UNDEFINED	**HMC Errors** PEPMAT or FPMAT. or CMPE. Specific PEMAT errors are reported in the PFHMC_ERRORINFO and PFHMC_ERRORDATA registers. Specific FPMAT errors are reported in the PFHMC_ERRORINFO_FPMAT and PFCHMC_ERRORDATA_FPMAT registers. Specific CMPE errors are reported in the PFCM_PE_CRCERRINFO.
PE_PUSH	27	0b	RCW	UNDEFINED	**PE Push** Indication that RDMA engine does not pull data from the push buffer (Indication that pe_pcie_push_rdy watchdog timer expired).
RESERVED	28	0b	RSV	N/A	Reserved.
VFLR	29	0b	RCW	UNDEFINED	**VFLR** VFLR was initiated by one of the VFs of the PF. The PF should read the GLGEN_VFLRSTAT getting an indication for the VF that generated the VFLR.
XLR_HW_DONE	30	0b	RCW	UNDEFINED	**XLR Hardware Done** VF or VM of the PF has set their hardware done indications after VM/VF reset.
SWINT	31	0b	RCW	UNDEFINED	**Software Interrupt indication**

13.2.2.15.32 PF Interrupt Other Cause Control - PFINT_OICR_CTL (0x0016CA80; RW)

Field definitions are the same as those defined in Section 13.2.2.15.12.

13.2.2.15.33 PF PE Asynchronous Event Queue Interrupt Cause Control - PFINT_AEQCTL (0x0016CB00; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MSIX_INDX	10:0	0x0	RW	UNDEFINED	**MSI-X Index** MSI-X vector index within the function space. The software should set the *MSIX_INDX* to values in the range of allocated interrupt vectors to the function.
ITR_INDX	12:11	00b	RW	UNDEFINED	**ITR Index** ITR Index of the interrupt cause. 00b = ITR0 01b = ITR1 10b = ITR2 11b = No ITR
RESERVED	29:13	0x0	RSV	N/A	Reserved.
CAUSE_ENA	30	0b	RW	UNDEFINED	**Cause Enable** Enable interrupt by this queue. When *CAUSE_ENA* is cleared, interrupts are not generated by the queue. The queue remains in the interrupt linked list and is processed at ITR expiration.
RESERVED	31	0b	RSV	N/A	Reserved.

13.2.2.15.34 PF0 Interrupt Other Cause CPM - PF0INT_OICR_CPM (0x0016CC40; RCW)

This register is set by the hardware with the same indication as the PFINT_OICR for the sake of the CPM driver.

Field definitions are the same as those defined in Section 13.2.2.15.31.

13.2.2.15.35 PF0 Interrupt Other Cause PSM - PF0INT_OICR_PSM (0x0016CC44; RCW)

This register is set by the hardware with the same indication as the PFINT_OICR for the sake of the PSM driver.

Field definitions are the same as those defined in Section 13.2.2.15.31.

13.2.2.15.36 PF0 Interrupt Other Cause CPM Control - PF0INT_OICR_CTL_CPM (0x0016CC48; RW)

This register is valid only for PF0. It has the same functionality of PFINT_OICR_CTL mapped in the same 4 KB page as PFINT_OICR3.

Field definitions are the same as those defined in Section 13.2.2.15.12.

13.2.2.15.37 PF0 Interrupt Other Cause HLP Enablement - PF0INT_OICR_ENA_HLP (0x0016CC4C; RW)

This register is valid only for PF0. It has the same functionality of PFINT_OICR_ENA mapped in the same 4 KB page as PFINT_OICR1.

Field definitions are the same as those defined in Section 13.2.2.15.29.

13.2.2.15.38 Global Interrupt TimeSync PHY Indication - GLINT_TSYN_PHY (0x0016CC50; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PHY_INDX	4:0	0x0	RW1C	DYNAMIC	**PHY Index** Bit 'i' in this register is set when Quad-PHY index 'i' initiates the transmit packet timestamp interrupt.
RESERVED	31:5	0x0	RSV	N/A	Reserved.

13.2.2.15.39 Global Interrupt Control - GLINT_CTL (0x0016CC54; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	15:0	0x0	RSV	N/A	Reserved.
ITR_GRAN_200	19:16	0x2	RW	UNDEFINED	**TITR Granularity 200** Defines the granularity of the ITR in 1 µs granularity for the 200 Gb/s Max speed SKU. Min allowed value is 1 µs.
ITR_GRAN_100	23:20	0x2	RW	UNDEFINED	**TITR Granularity 100** Defines the granularity of the ITR in 1 µs granularity for the 100 Gb/s Max speed SKU. Min allowed value is 1 µs.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ITR_GRAN_50	27:24	0x2	RW	UNDEFINED	**TITR Granularity 50** Defines the granularity of the ITR in 1 µs granularity for the 50 Gb/s Max speed SKU. Min allowed value is 2 µs.
ITR_GRAN_25	31:28	0x4	RW	UNDEFINED	**TITR Granularity 25** Defines the granularity of the ITR in 1 µs granularity for the 25 Gb/s Max speed SKU. Min allowed value is 4 µs.

13.2.2.15.40 PF0 Interrupt Other Cause PSM Enablement - PF0INT_OICR_ENA_PSM (0x0016CC58; RW)

This register is valid only for PF0. It has the same functionality of PFINT_OICR_ENA mapped in the same 4 KB page as PFINT_OICR2.

Field definitions are the same as those defined in Section 13.2.2.15.29.

13.2.2.15.41 PF0 Interrupt Other Cause HLP Control - PF0INT_OICR_CTL_HLP (0x0016CC5C; RW)

This register is valid only for PF0. It has the same functionality of PFINT_OICR_CTL mapped in the same 4 KB page as PFINT_OICR1.

Field definitions are the same as those defined in Section 13.2.2.15.12.

13.2.2.15.42 PF0 Interrupt Other Cause CPM Enablement - PF0INT_OICR_ENA_CPM (0x0016CC60; RW)

This register is valid only for PF0. It has the same functionality of PFINT_OICR_ENA mapped in the same 4 KB page as PFINT_OICR3.

Field definitions are the same as those defined in Section 13.2.2.15.29.

13.2.2.15.43 PF0 Interrupt Other Cause PSM Control - PF0INT_OICR_CTL_PSM (0x0016CC64; RW)

This register is valid only for PF0. It has the same functionality of PFINT_OICR_CTL mapped in the same 4 KB page as PFINT_OICR2.

Field definitions are the same as those defined in Section 13.2.2.15.12.

13.2.2.15.44 PF0 Interrupt Other Cause HLP - PF0INT_OICR_HLP (0x0016CC68; RCW)

This register is set by the hardware with the same indication as the PFINT_OICR for the sake of the HLP driver.

Field definitions are the same as those defined in Section 13.2.2.15.31.

13.2.2.15.45 Global Interrupt TimeSync Master Select - GLINT_TSYN_PFMSTR[n] (0x0016CCC0 + 0x4*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PF_MASTER	2:0	000b	RW	UNDEFINED	**PF Master** This field indicates the index of the PF that owns the 1588 master timer 'n', where 'n' is the register index.
RESERVED	31:3	0x0	RSV	N/A	Reserved.

13.2.2.15.46 VF Vector Allocation - VPINT_ALLOC[VF] (0x001D1000 + 0x4*VF, VF=0...255; RW)

This register indicates the interrupt allocation of the VF.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FIRST	10:0	0x0	RW	UNDEFINED	**First** Index of the first interrupt vector of the VF in the internal physical space.
RESERVED	11	0b	RSV	N/A	Reserved.
LAST	22:12	0x0	RW	UNDEFINED	**Last** Index of the last interrupt vector of the VF in the internal physical space.
RESERVED	30:23	0x0	RSV	N/A	Reserved.
VALID	31	0b	RW	UNDEFINED	**Valid** Valid indication. When this bit is clear, the contents of this CSR are not valid.

13.2.2.15.47 PF Vector Allocation - PFINT_ALLOC (0x001D2600; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FIRST	10:0	0x0	RW	UNDEFINED	**First** Index of the first interrupt vector of the PF in the internal physical space.
RESERVED	11	0b	RSV	N/A	Reserved.
LAST	22:12	0x0	RW	UNDEFINED	**Last** Index of the last interrupt vector of the PF and its VFs in the internal physical space.
RESERVED	30:23	0x0	RSV	N/A	Reserved.
VALID	31	0b	RW	UNDEFINED	**Valid** Valid indication. When this bit is clear, the contents of this CSR are not valid.

13.2.2.16 PF - Virtualization PF Registers

13.2.2.16.1 PF Resources Allocation - PF_VT_PFALLOC_HIF (0x0009DD80; RW)

This register indicates the VFs allocated to the each PF.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FIRSTVF	7:0	0x0	RW	UNDEFINED	**First VF** The first VF allocated to this PF. Valid only if the *VALID* flag is set. Valid values are 0-255.
LASTVF	15:8	0x0	RW	UNDEFINED	**Last VF** The last VF allocated to this PF. Valid only if the *VALID* flag is set. Valid values are 0-255.
RESERVED	30:16	0x0	RSV	N/A	Reserved.
VALID	31	0b	RW	UNDEFINED	**Valid** The *FIRSTVF* and *LASTVF* fields in this register are valid. If cleared no VFs are allocated to this PF. If cleared, the SR-IOV capability should not be exposed for this PF.

13.2.2.16.2 PF Virtualization Status Register - PF_VIRT_VSTATUS (0x0009E680; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
NUM_VFS	7:0	0x0	RO	N/A	**Number of VFs** Reflects the value of the *Num VFs* field in the IOV capability structure.
TOTAL_VFS	15:8	0x0	RO	N/A	**Total VFs** Reflects the value of the *TotalVFs* field in the IOV capability structure.
IOV_ACTIVE	16	0b	RO	N/A	**IOV Active** Reflects the value of the VF Enable (*VFE*) bit in the IOV control/status register
RESERVED	31:17	0x0	RSV	N/A	Reserved.

13.2.2.16.3 PF Resources Allocation - PF_VT_PFALLOC_PCIE (0x000BE080; RW)

This register indicates the VFs allocated to the each PF.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FIRSTVF	7:0	0x0	RW	UNDEFINED	**First VF** The first VF allocated to this PF. Valid only if the *VALID* flag is set. Valid values are 0-255.
LASTVF	15:8	0x0	RW	UNDEFINED	**Last VF** The last VF allocated to this PF. Valid only if the *VALID* flag is set. Valid values are 0-255.
RESERVED	30:16	0x0	RSV	N/A	Reserved.
VALID	31	0b	RW	UNDEFINED	**Valid** The *FIRSTVF* and *LASTVF* fields in this register are valid. If cleared no VFs are allocated to this PF. If cleared, the SR-IOV capability should not be exposed for this PF.

13.2.2.16.4 PF Resources Allocation - PF_VT_PFALLOC (0x001D2480; RW)

This register indicates the VFs allocated to the each PF.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FIRSTVF	7:0	0x0	RW	UNDEFINED	**First VF** The first VF allocated to this PF. Valid only if the *VALID* flag is set. Valid values are 0-255.
LASTVF	15:8	0x0	RW	UNDEFINED	**Last VF** The last VF allocated to this PF. Valid only if the *VALID* flag is set. Valid values are 0-255.
RESERVED	30:16	0x0	RSV	N/A	Reserved.
VALID	31	0b	RW	UNDEFINED	**Valid** The *FIRSTVF* and *LASTVF* fields in this register are valid. If cleared no VFs are allocated to this PF. If cleared, the SR-IOV capability should not be exposed for this PF.

13.2.2.17 PF - DCB Registers

13.2.2.17.1 DCB TDPU Control - PRTDCB_TDPUC (0x00040940; RW)

Malicious indication/control register. Drop reasons are not valid on markers/head update packets.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MAX_TXFRAME	15:0	0x2600	RW	UNDEFINED	**Max Tx Frame** Maximum Tx frame size in bytes. Frames longer than the size specified here are discarded and not transmitted (nor looped back). Default and maximum allowed value is 9.5 KB (jumbo frame). Minimum size to be configured here is 60.
MAL_LENGTH	16	0b	RO	N/A	**Malicious Length** Detected malicious length parameters in the Tx-Descriptor
MAL_CMD	17	0b	RO	N/A	**Malicious Commands** Detected malicious combination of commands vs. packet types in the Tx-Descriptor.
TTL_DROP	18	0b	RO	N/A	**TTL Drop** Packet was dropped due to TTL parameter. This indication is invalid on *DUMMY* or UR_DROP.
UR_DROP	19	0b	RO	N/A	**Unsupported Requests Drop** Packet dropped due to unsupported request or dummy completion event of requested packet data.
DUMMY	20	0b	RO	N/A	**Dummy** Packet was dropped because dummy bit was on
BIG_PKT_SIZE	21	0b	RO	N/A	**Big Packet Size** Packet dropped because it exceeds *MAX_TXFRAME* size.
L2_ACCEPT_FAIL	22	0b	RO	N/A	**L2 Acceptance Fail** Packet dropped because of L2 acceptance rules fail. This indication is invalid on *DUMMY* or *UR_DROP*.
DSCP_CHECK_FAIL	23	0b	RO	N/A	**DSCP Check Fail** Packet dropped because of DSCP rule check fail.
RCU_ANTISPOOF	24	0b	RO	N/A	**RCU Anti-Spoof** Packet dropped by RCU.
NIC_DSI	25	0b	RO	N/A	**NIC DSI** Packet dropped because DSI command in NIC mode was identified.
NIC_IPSEC	26	0b	RO	N/A	**NIC IPsec** Packet dropped because IPSEC command in NIC mode was identified.
RESERVED	30:27	0x0	RSV	N/A	Reserved.
CLEAR_DROP	31	0b	SC	UNDEFINED	**Clear Drop** Setting this bit, clears the malicious and drop flags in this register. This bit is auto-cleared.

13.2.2.17.2 DCB Receive UP in TDPU - PRTDCB_RUP_TDPU (0x00040960; RW)

Default UP per port.

Field	Bit(s)	Init.	Type	CFG Policy	Description
NOVLANUP	2:0	000b	RW	UNDEFINED	**No VLAN UP** This field assigns a default UP value to untagged incoming packets. The default UP is not inserted in the packet itself, but it controls in which linked list of RPB untagged packets are stored. UP 0 is the default.
RESERVED	31:3	0x0	RSV	N/A	Reserved.

13.2.2.17.3 Tx DCB DSCP to User Priority Control - PRTDCB_TX_DSCP2UP_CTL (0x00040980; RW)

DSCP mode enable/disable per port:

Field	Bit(s)	Init.	Type	CFG Policy	Description
DSCP2UP_ENA	0	0b	RW	UNDEFINED	**DSCP-to-UP Enable** This flag enables the DSCP to User Priority Lookup table. 0b = The DSCP lookup table is disabled and the User Priority is taken from the VLAN tag. 1b = The DSCP to User Priority LUP is enabled.
DSCP_DEFAULT_UP	3:1	000b	RW	UNDEFINED	**DSCP Default UP** Default User Priority for packets without an IP header. This field is meaningful only when the DSCP2UP LUT is enabled by the *DSCP2UP_ENA* flag in this register.
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.17.4 Tx DCB DSCP to User Priority LUT for IPv4 Packets - PRTDCB_TX_DSCP2UP_IPV4_LUT[n] (0x000409A0 + 0x20*n, n=0...7; RW)

This register is used to load Tx DSCP2UP tables for IPv4 packets. The table is used for DSCP to UP manipulation, for ports which are DSCP enabled. Result UP from the table is passed in metadata for loopback packets.

Field definitions are the same as those defined in Section 13.2.2.17.5.

13.2.2.17.5 Tx DCB DSCP to User Priority LUT for IPv6 Packets - PRTDCB_TX_DSCP2UP_IPV6_LUT[n] (0x00040AA0 + 0x20*n, n=0...7; RW)

This register is used to load Tx DSCP2UP tables for IPv6 packets. The table is used for DSCP to UP manipulation, for ports which are DSCP enabled. Result UP from the table is passed in metadata for loopback packets.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DSCP2UP_LUT_0	2:0	000b	RW	UNDEFINED	**DSCP-to-UP LUT 0** This entry in register 'n' of the array represents the User Priority output for DSCP input equals to '8*n'.
RESERVED	3	0b	RSV	N/A	Reserved.
DSCP2UP_LUT_1	6:4	000b	RW	UNDEFINED	**DSCP-to-UP LUT 1** This entry in register 'n' of the array represents the User Priority output for DSCP input equals to '8*n+1'.
RESERVED	7	0b	RSV	N/A	Reserved.
DSCP2UP_LUT_2	10:8	000b	RW	UNDEFINED	**DSCP-to-UP LUT 2** This entry in register 'n' of the array represents the User Priority output for DSCP input equals to '8*n+2'.
RESERVED	11	0b	RSV	N/A	Reserved.
DSCP2UP_LUT_3	14:12	000b	RW	UNDEFINED	**DSCP-to-UP LUT 3** This entry in register 'n' of the array represents the User Priority output for DSCP input equals to '8*n+3'.
RESERVED	15	0b	RSV	N/A	Reserved.
DSCP2UP_LUT_4	18:16	000b	RW	UNDEFINED	**DSCP-to-UP LUT 4** This entry in register 'n' of the array represents the User Priority output for DSCP input equals to '8*n+4'.
RESERVED	19	0b	RSV	N/A	Reserved.
DSCP2UP_LUT_5	22:20	000b	RW	UNDEFINED	**DSCP-to-UP LUT 5** This entry in register 'n' of the array represents the User Priority output for DSCP input equals to '8*n+5'.
RESERVED	23	0b	RSV	N/A	Reserved.
DSCP2UP_LUT_6	26:24	000b	RW	UNDEFINED	**DSCP-to-UP LUT 6** This entry in register 'n' of the array represents the User Priority output for DSCP input equals to '8*n+6'.
RESERVED	27	0b	RSV	N/A	Reserved.
DSCP2UP_LUT_7	30:28	000b	RW	UNDEFINED	**DSCP-to-UP LUT 7** This entry in register 'n' of the array represents the User Priority output for DSCP input equals to '8*n+7'.
RESERVED	31	0b	RSV	N/A	Reserved.

13.2.2.17.6 Transmit DCSP to TC Enforcement - IPv4 - GL_DCB_TDSCP2TC_BLOCK_IPV4[n] (0x00049018 + 0x4*n, n=0...63; RO)

Defines DSCP-to-TC enforcement for IPv4 packets.

For each DSCP (0..63) this register maps the TCs for which this DSCP is allowed, by having corresponding *TC* bit set or cleared.

For DSCP #n, CSR contains 32 entries. Each entry represents TC per port, according to device mode of operation:

- 1 port x 32 TCs: bit[N] represents TC[N]
- 2 ports x 8 TCs: bit[N] represents port[P] x TC[T], where:

  P = 0..1
  T = 0..7
  N = P*8 + T
  bits 16..31 are not valid
- 4 ports x 8 TCs: bit[N] represents port[P] x TC[T], where:

  P = 0..4
  T = 0..7
  N = P*8 + T
- 8 ports x 4 TCs: bit[N] represents port[P] x TC[T], where:

  P = 0..7
  T = 0..3
  N = P*4 + T

For each entry:

0b = DSCP is allowed for this TC (packet is not blocked)
1b = DSCP is blocked for this TC (packet is blocked if enabled by DSCP2TC_BLOCK_DIS register).

Field	Bit(s)	Init.	Type	CFG Policy	Description
TC_BLOCK_LUT	31:0	0x0	RW	UNDEFINED	**TC Block LUT** Defines if it is allowed to have DSCP #N in packet that is sent through port=x, tc=y and port is DSCP enabled. If number of ports = X, number of TCs = Y, bit Y*port + TC defines if DSCP is allowed for this port/TC. 0b = Allowed. 1b = Block (if enabled by DSCP2TC_BLOCK_DIS register).

13.2.2.17.7 Transmit DCSP to TC Enforcement - IPv6 - GL_DCB_TDSCP2TC_BLOCK_IPV6[n] (0x00049118 + 0x4*n, n=0...63; RO)

Defines DSCP-to-TC enforcement for IPv6 packets.

For each DSCP (0..63) this register maps the TCs for which this DSCP is allowed, by having corresponding *TC* bit set or cleared.

For DSCP #n, CSR contains 32 entries. Each entry represents TC per port, according to device mode of operation:

- 1 port x 32 TCs: bit[N] represents TC[N]
- 2 ports x 8 TCs: bit[N] represents port[P] x TC[T], where:

  P = 0..1
  T = 0..7
  N = P*8 + T
  bits 16..31 are not valid
- 4 ports x 8 TCs: bit[N] represents port[P] x TC[T], where:

  P = 0..4
  T = 0..7
  N = P*8 + T
- 8 ports x 4 TCs: bit[N] represents port[P] x TC[T], where:

  P = 0..7
  T = 0..3
  N = P*4 + T

For each entry:

0b = DSCP is allowed for this TC (packet is not blocked)
1b = DSCP is blocked for this TC (packet is blocked if enabled by DSCP2TC_BLOCK_DIS register).

Field definitions are the same as those defined in Section 13.2.2.17.6.

13.2.2.17.8 Transmit DCSP to TC Enforcement Enable - GL_DCB_TDSCP2TC_BLOCK_DIS (0x00049218; RO)

If a packet failed in DSCP-to-TC enforcement checks (as configured by TDSCP2TC_BLOCK LUTs), this register determines if the packet is dropped or not.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DSCP2TC_BLOCK_DIS	0	1b	RW	UNDEFINED	**DSCP-to-TC Block Disable** 0b = If DSCP2TC LUT returned status = FAIL, packet is dropped. 1b = Packet is not dropped due to DSCP2TC LUT status=FAIL.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.17.9 Port DCB General Control - PRTDCB_GENC (0x00083000; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	1:0	00b	RSV	N/A	Reserved.
NUMTC	5:2	0x1	RW	UNDEFINED	**Number of TCs** Number of Traffic Classes (TCs) for the port. This field must be set consistently with the settings made in the Tx-Scheduler.
FCOEUP	8:6	0x3	RW	UNDEFINED	Defines the 802.1p UP field used for FCoE traffic over the link.
FCOEUP_VALID	9	1b	RW	UNDEFINED	**FCoE UP Valid** Validity bit for the *FCoEUP* field. 0b = FCoE is not used over this port, and therefore RPB settings must use the same Max Frame Size for all TCs. 1b = The *FCoEUP* field contents is valid.
RESERVED	15:10	0x0	RSV	N/A	Reserved.
PFCLDA	31:16	0x079D	RW	UNDEFINED	**PFC Link Delay Allowance** It is expressed in 16-bytes unit. 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 must be 0x1E70.

13.2.2.17.10 Port DCB General Status - PRTDCB_GENS (0x00083020; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DCBX_STATUS	2:0	000b	RW	N/A	**DCBX Status** 000b = NOT_STARTED 001b = IN_PROGRESS 010b = DONE 011b = MULTIPLE_PEERS 111b = DISABLED All other values are reserved.
RESERVED	31:3	0x0	RSV	N/A	Reserved

13.2.2.17.11 Global DCB General Control - GLDCB_GENC (0x00083044; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PCIRTT	15:0	0x009C	RW	UNDEFINED	**PCIe Round Trip Time** It is expressed in 16-bytes unit. Default is 2 ms PCIe round trip time assuming 10G links (no optimization done for lower links speeds). For a 40G link, the number must be 0x0270.
RESERVED	31:16	0x0	RSV	N/A	

13.2.2.17.12 DCB Transmit Port DWRR Status - TPB_PRTDCB_TCB_DWRR_CREDITS (0x000991C0; RO)

Field definitions are the same as those defined in Section 13.2.2.17.47.

13.2.2.17.13 DCB Transmit Port DWRR Quanta/Weights - TPB_PRTDCB_TCB_DWRR_QUANTA (0x00099220; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
QUANTA	10:0	0x1	RW	UNDEFINED	**Quanta** Port quanta size in 64-byte granularity. Actual bandwidth share is determined by considering the other ports' quantas.
RESERVED	31:11	0x0	RSV	N/A	Reserved.

13.2.2.17.14 DCB Transmit Port DWRR Saturation Value - TPB_PRTDCB_TCB_DWRR_SAT (0x00099260; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SATURATION	16:0	0x1000	RW	UNDEFINED	**Saturation** Port saturation value in bytes. Port DWRR credits cannot be above this value.
RESERVED	31:17	0x0	RSV	N/A	Reserved.

13.2.2.17.15 DCB Transmit Regular Bulk DWRR Status - TPB_PRTTCB_BULK_DWRR_REG_CREDITS (0x000992A0; RO)

Field definitions are the same as those defined in Section 13.2.2.17.50.

13.2.2.17.16 DCB Transmit Wait Bulk DWRR Status - TPB_PRTTCB_BULK_DWRR_WB_CREDITS (0x000992C0; RO)

Field definitions are the same as those defined in Section 13.2.2.17.51.

13.2.2.17.17 DCB Transmit Regular Low Latency DWRR Status - TPB_PRTTCB_LL_DWRR_REG_CREDITS (0x00099300; RO)

Field definitions are the same as those defined in Section 13.2.2.17.52.

13.2.2.17.18 DCB Transmit Wait Low Latency DWRR Status - TPB_PRTTCB_LL_DWRR_WB_CREDITS (0x00099320; RO)

Field definitions are the same as those defined in Section 13.2.2.17.53.

13.2.2.17.19 DCB Transmit Regular Bulk DWRR Quanta/Weights - TPB_BULK_DWRR_REG_QUANTA (0x00099340; RW)

Field definitions are the same as those defined in Section 13.2.2.17.54.

13.2.2.17.20 DCB Transmit Wait Bulk DWRR Quanta/Weights - TPB_BULK_DWRR_WB_QUANTA (0x00099344; RW)

Field definitions are the same as those defined in Section 13.2.2.17.55.

13.2.2.17.21 DCB Transmit Regular Low Latency DWRR Quanta/Weights - TPB_LL_DWRR_REG_QUANTA (0x00099348; RW)

Field definitions are the same as those defined in Section 13.2.2.17.56.

13.2.2.17.22 DCB Transmit Wait Low Latency DWRR Quanta/Weights - TPB_LL_DWRR_WB_QUANTA (0x0009934C; RW)

Field definitions are the same as those defined in Section 13.2.2.17.57.

13.2.2.17.23 DCB Transmit Regular Bulk DWRR Saturation Value - TPB_BULK_DWRR_REG_SAT (0x00099350; RW)

Field definitions are the same as those defined in Section 13.2.2.17.58.

13.2.2.17.24 DCB Transmit Wait Bulk DWRR Saturation Value - TPB_BULK_DWRR_WB_SAT (0x00099354; RW)

Field definitions are the same as those defined in Section 13.2.2.17.59.

13.2.2.17.25 DCB Transmit Regular Low Latency DWRR Saturation Value - TPB_LL_DWRR_REG_SAT (0x00099358; RW)

Field definitions are the same as those defined in Section 13.2.2.17.60.

13.2.2.17.26 DCB Transmit Wait Low Latency DWRR Saturation Value - TPB_LL_DWRR_WB_SAT (0x0009935C; RW)

Field definitions are the same as those defined in Section 13.2.2.17.61.

13.2.2.17.27 DCB Transmit Rate Limiter Control per TC - TPB_WB_RL_TC_CFG[n] (0x00099360 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.17.66.

13.2.2.17.28 DCB Transmit Rate Limiter Status per TC - TPB_WB_RL_TC_STAT[n] (0x000993E0 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.17.67.

13.2.2.17.29 TC Rate Limiters Config - GLTPB_WB_RL (0x00099460; RO)

Field definitions are the same as those defined in Section 13.2.2.17.68.

13.2.2.17.30 TPB TC LL Config - GLDCB_TPB_TCLL_CFG (0x00099464; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LLTC	31:0	0x0	RW	UNDEFINED	**Low Latency TC** 1-bit entry per each TC. Each entry controls whether the TC is considered to have low latency needs for the transmit path. 0b = TC is defined to be a Bulk. 1b = TC is defined to be a Low Latency TC for transmit.

13.2.2.17.31 TPB TLPM TC Immediate FC Enable - GLDCB_TPB_IMM_TLPM (0x00099468; RO)

Configures FC immediate mode per TC for TCUPM interface.

Field	Bit(s)	Init.	Type	CFG Policy	Description
IMM_EN	31:0	0xFFFFFFFF	RW	UNDEFINED	**Immediate Enable** Per-TC Tx immediate FC enable. 0b = Conditioned XOFF Forwarding Mode (default). XOFF notifications received from the line or from the internal loopback path are not forwarded upward to the TCUPM, but only once the data pipe monitor for this TC is filled over its threshold. This mode provides better XON recovering time. 1b = Immediate XOFF Forwarding Mode. XOFF notifications received from the line or from the internal loopback path are immediately forwarded internally upward to the TCUPM. This mode is useful to support PFC-enabled TCs of the port like true independent traffic classes when more than two such TCs are configured over the port. This bit should be kept as zero

13.2.2.17.32 TPB TC Immediate FC Enable - GLDCB_TPB_IMM_TPB (0x0009946C; RO)

Configures FC immediate mode per TC for TCB interface.

Field	Bit(s)	Init.	Type	CFG Policy	Description
IMM_EN	31:0	0xFFFFFFFF	RW	UNDEFINED	**Immediate Enable** Per-TC Tx immediate FC enable. 0b = Conditioned XOFF Forwarding Mode (default). XOFF notifications received from the line or from the internal loopback path are not forwarded upward to the TCB, but only once the data pipe monitor for this TC is filled over its threshold. This mode provides better XON recovering time. 1b = Immediate XOFF Forwarding Mode. XOFF notifications received from the line or from the internal loopback path are immediately forwarded internally upward to the TCB.

13.2.2.17.33 Ignore FC per TC List - GLDCB_TFPFCI (0x0009949C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GLDCB_TFPFCI	31:0	0x0	RW	UNDEFINED	A bit per TC, that if high FC is ignored for this TC: Packets are dropped and not sent to MAC/RPB (even if FC from MAC/RPB is not set). Indication to pipe monitor is not asserted. MAC/RPB FC is ignored.

13.2.2.17.34 TCB Arbiter Credit Expansion - TPB_PRTTCB_CREDIT_EXP (0x00099644; RW)

Register used to add arbiter credit expansion for packets per port.

Field definitions are the same as those defined in Section 13.2.2.17.62.

13.2.2.17.35 TCB Arbiter Credit Expansion Control - TPB_GLTCB_CREDIT_EXP_CTL (0x00099664; RW)

Used to enable/disable arbiter credit expansion and set minimal packet size for updating credits.

Field definitions are the same as those defined in Section 13.2.2.17.63.

13.2.2.17.36 Global Wait Buffer Strict Priority Enable - TPB_GLDCB_TCB_WB_SP (0x0009966C; RW)

Field definitions are the same as those defined in Section 13.2.2.17.69.

13.2.2.17.37 DCB Transmit Data Pipe Port Monitor Status - PRTDCB_TLPM_REG_DM (0x000A0000; RO)

Regular buffer data monitor status per port.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MONITOR	18:0	0x0	RW	UNDEFINED	**Monitor** Regular buffer port monitor status. Current amount of data in lower pipe not including TPB wait buffer per port, in bytes.
RESERVED	31:19	0x0	RSV	N/A	Reserved.

13.2.2.17.38 DCB Transmit Data Pipe Port Monitor Threshold - PRTDCB_TLPM_REG_DTHR (0x000A0020; RW)

Regular buffer data monitor threshold per port.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PORTOFFTH_H	11:0	0x1A9	RW	UNDEFINED	**Port OFF Threshold High** Regular buffer port monitor high threshold applied over the Data Pipe. It is expressed in 128-byte units of Layer2 packet lengths (i.e., excluding preamble, IPG, and CRC). When over this threshold, monitor blocks the port.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PORTOFFTH_L	23:12	0x1A9	RW	UNDEFINED	**Port OFF Threshold Low** Regular buffer port monitor low threshold applied over the Data Pipe. It is expressed in 128Byte units of Layer2 packet lengths (i.e., excluding preamble, IPG, and CRC). When over this threshold, monitor blocks bulk traffic of the port.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.17.39 DCB Transmit Data Pipe Port Waiting Monitor Status - PRTDCB_TLPM_WAIT_PFC_DM (0x000A0040; RO)

Wait + regular buffer data monitor status per port of PFC-enabled TCs only.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MONITOR	18:0	0x0	RW	UNDEFINED	**Monitor** Wait buffer port monitor status. Current amount of data in lower pipe for both regular and wait buffers of TPB per port, in bytes.
RESERVED	31:19	0x0	RSV	N/A	Reserved.

13.2.2.17.40 DCB Transmit Data Pipe Port Waiting Monitor Threshold - PRTDCB_TLPM_WAIT_PFC_DTHR (0x000A0060; RW)

Wait + regular buffer data monitor threshold per port.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PORTOFFTH	11:0	0x3A0	RW	UNDEFINED	**Port OFF Threshold** High threshold on the total amount of data that is waiting in the TPB waiting lists of the port for all its PFC-enabled TCs, in 128-byte units.
RESERVED	31:12	0x0	RSV	N/A	Reserved.

13.2.2.17.41 DCB Transmit Data Pipe TC Waiting Monitor Status - TCDCB_TLPM_WAIT_DM[n] (0x000A0080 + 0x4*n, n=0...31; RO)

Wait + regular buffer data monitor status per TC.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MONITOR	18:0	0x0	RW	UNDEFINED	**Monitor** Wait buffer TC monitor status. Current amount of data in lower pipe for both regular and wait buffers of TPB per TC (32), in bytes.
RESERVED	31:19	0x0	RSV	N/A	Reserved.

13.2.2.17.42 DCB Transmit Data Pipe TC Waiting Monitor Threshold - TCDCB_TLPM_WAIT_DTHR[n] (0x000A0100 + 0x4*n, n=0...31; RW)

Wait + regular buffer data monitor threshold per TC.

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCOFFTH	11:0	0x1A9	RW	UNDEFINED	**TC Off Threshold** The total amount of data that can accumulate in the TPB (waiting list included) of the port for the same PFC-enabled TC, in 128B units.
RESERVED	31:12	0x0	RSV	N/A	Reserved.

13.2.2.17.43 DCB PCIe Tx Data Count - GLDCB_TLPM_PCI_DM (0x000A0180; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MONITOR	18:0	0x0	RW	UNDEFINED	**Monitor** Byte count of all the Tx data (for all ports) which is in transit from host PCIe to TDPU block.
RESERVED	31:19	0x0	RSV	N/A	Reserved.

13.2.2.17.44 DCB PCIe Tx Data Threshold - GLDCB_TLPM_PCI_DTHR (0x000A0184; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PCI_TDATA	11:0	0x96	RW	UNDEFINED	**PCI Data Threshold** Shared threshold expressed in 128-byte units for all Tx data (from all ports) that is in transit between host and TDPU block. When the threshold is reached, no Tx data read request is sent to the host. When we pass under the threshold, data read request(s) for an entire frame can be sent to the host, and so on until we reach the threshold again. The byte count register associated with this threshold is GLDCB_TLPM_PCI_DATA. Default of 19200 bytes provides some good margin when assuming a PCIe round trip time of 1.5 µs.
RESERVED	31:12	0x0	RSV	N/A	Reserved.

13.2.2.17.45 DCB TC Immediate FC Enable - GLDCB_TLPM_IMM_TCUPM (0x000A018C; RW)

Configures FC immediate mode per TC for TCUPM interface.

Field	Bit(s)	Init.	Type	CFG Policy	Description
IMM_EN	31:0	0xFFFFFFFF	RW	UNDEFINED	**Immediate Enable** Per-TC Tx immediate FC enable. 0b = Conditioned XOFF Forwarding Mode (default). XOFF notifications received from the line or from the internal loopback path are not forwarded upward to the TCUPM, but only once the data pipe monitor for this TC is filled over its threshold. This mode provides better XON recovering time. 1b = Immediate XOFF Forwarding Mode. XOFF notifications received from the line or from the internal loopback path are immediately forwarded internally upward to the TCUPM. This mode is useful to support PFC-enabled TCs of the port like true independent traffic classes when more than two such TCs are configured over the port. This bit should be kept as zero

13.2.2.17.46 DCB TC Immediate FC Mode - GLDCB_TLPM_IMM_TCB (0x000A0190; RW)

Configures FC immediate mode per TC for TCB interface.

Field	Bit(s)	Init.	Type	CFG Policy	Description
IMM_EN	31:0	0xFFFFFFFF	RW	UNDEFINED	**Immediate Enable** Per-TC Tx immediate FC enable. 0b = Conditioned XOFF Forwarding Mode (default). XOFF notifications received from the line or from the internal loopback path are not forwarded upward to the TCB, but only once the data pipe monitor for this TC is filled over its threshold. This mode provides better XON recovering time. 1b = Immediate XOFF Forwarding Mode. XOFF notifications received from the line or from the internal loopback path are immediately forwarded internally upward to the TCB.

13.2.2.17.47 DCB Transmit Port DWRR Status - PRTDCB_TCB_DWRR_CREDITS (0x000AE000; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CREDITS	17:0	0x0	RW	UNDEFINED	**Credits** Current amount of credits accumulated by the port in Tx. The amount is expressed in byte units and it is formatted as an algebraic 2's complement number. Writing to this field has the effect of loading a new current credit value, which is used by Tx-PRR algorithm.
RESERVED	31:18	0x0	RSV	N/A	Reserved.

13.2.2.17.48 DCB Transmit Port DWRR Quanta/Weights - PRTDCB_TCB_DWRR_QUANTA (0x000AE020; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
QUANTA	10:0	0x14	RW	UNDEFINED	**Quanta** Port quanta size in 64-byte granularity. Actual bandwidth share is determined by considering the ratio with other ports' quantas.
RESERVED	31:11	0x0	RSV	N/A	Reserved.

13.2.2.17.49 DCB Transmit Port DWRR Saturation Value - PRTDCB_TCB_DWRR_SAT (0x000AE040; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SATURATION	16:0	0x3000	RW	UNDEFINED	**Saturation** Port saturation value in bytes. Port DWRR credits cannot be above this value. Minimal Saturation value depends on Port MTU Size (for all quantas): MTU is [0k-2k]: 3K MTU is [2k-4k]: 6K MTU is [4k-8k]: 12K MTU is [8k-16k] (Jumbo): 24K Final Minimal Saturation = Max(Quanta, MTU Min Saturation).
RESERVED	31:17	0x0	RSV	N/A	Reserved.

13.2.2.17.50 DCB Transmit Regular Bulk DWRR Status - PRTTCB_BULK_DWRR_REG_CREDITS (0x000AE060; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CREDITS	17:0	0x0	RW	UNDEFINED	**Credits** Current amount of credits accumulated by the port for bulk regular buffer in Tx. The amount is expressed in byte units and it is formatted as an algebraic 2's complement number. Writing to this field has the effect of loading a new current credit value.
RESERVED	31:18	0x0	RSV	N/A	Reserved.

13.2.2.17.51 DCB Transmit Wait Bulk DWRR Status - PRTTCB_BULK_DWRR_WB_CREDITS (0x000AE080; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CREDITS	17:0	0x0	RW	UNDEFINED	**Credits** Current amount of credits accumulated by the port for bulk wait buffer in Tx. The amount is expressed in byte units and it is formatted as an algebraic 2's complement number. Writing to this field has the effect of loading a new current credit value.
RESERVED	31:18	0x0	RSV	N/A	Reserved.

13.2.2.17.52 DCB Transmit Regular Low Latency DWRR Status - PRTTCB_LL_DWRR_REG_CREDITS (0x000AE0A0; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CREDITS	17:0	0x0	RW	UNDEFINED	**Credits** Current amount of credits accumulated by the port for low latency regular buffer in Tx. The amount is expressed in byte units and it is formatted as an algebraic 2's complement number. Writing to this field has the effect of loading a new current credit value.
RESERVED	31:18	0x0	RSV	N/A	Reserved.

13.2.2.17.53 DCB Transmit Wait Low Latency DWRR Status - PRTTCB_LL_DWRR_WB_CREDITS (0x000AE0C0; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CREDITS	17:0	0x0	RW	UNDEFINED	**Credits** Current amount of credits accumulated by the port for low latency wait buffer in Tx. The amount is expressed in byte units and it is formatted as an algebraic 2's complement number. Writing to this field has the effect of loading a new current credit value.
RESERVED	31:18	0x0	RSV	N/A	Reserved.

13.2.2.17.54 DCB Transmit Regular Bulk DWRR Quanta/Weights - GLTCB_BULK_DWRR_REG_QUANTA (0x000AE0E0; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
QUANTA	10:0	0x1	RW	UNDEFINED	**Quanta** Quanta for bulk buffer DWRR arbitration in 64-byte granularity. This quanta is used for all ports. Actual bandwidth share is determined by considering the ratio with the wait buffer's bulk quanta.
RESERVED	31:11	0x0	RSV	N/A	Reserved.

13.2.2.17.55 DCB Transmit Wait Bulk DWRR Quanta/Weights - GLTCB_BULK_DWRR_WB_QUANTA (0x000AE0E4; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
QUANTA	10:0	0x2	RW	UNDEFINED	**Quanta** Quanta for bulk wait buffer DWRR arbitration in 64-byte granularity. This quanta is used for all ports. Actual bandwidth share is determined by considering the ratio with the regular buffer's bulk quanta.
RESERVED	31:11	0x0	RSV	N/A	Reserved.

13.2.2.17.56 DCB Transmit Regular Low Latency DWRR Quanta/Weights - GLTCB_LL_DWRR_REG_QUANTA (0x000AE0E8; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
QUANTA	10:0	0x1	RW	UNDEFINED	**Quanta** Quanta for low latency regular buffer DWRR arbitration in 64-byte granularity. Actual bandwidth share is determined by considering the ratio with the wait buffer's low latency quanta.
RESERVED	31:11	0x0	RSV	N/A	Reserved.

13.2.2.17.57 DCB Transmit Wait Low Latency DWRR Quanta/Weights - GLTCB_LL_DWRR_WB_QUANTA (0x000AE0EC; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
QUANTA	10:0	0x2	RW	UNDEFINED	**Quanta** Quanta for low latency wait buffer DWRR arbitration in 64-byte granularity. Actual bandwidth share is determined by considering the ratio with the regular buffer's low latency quanta.
RESERVED	31:11	0x0	RSV	N/A	Reserved.

13.2.2.17.58 DCB Transmit Regular Bulk DWRR Saturation Value - GLTCB_BULK_DWRR_REG_SAT (0x000AE0F0; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SATURATION	16:0	0x2000	RW	UNDEFINED	**Saturation** Bulk regular buffer saturation value in bytes for all ports. Credits cannot be added over this value. Minimal Saturation value depends on Port MTU Size (for all quantas): MTU is [0k-2k]: 3K MTU is [2k-4k]: 6K MTU is [4k-8k]: 12K MTU is [8k-16k] (Jumbo): 24K Final Minimal Saturation = Max(Quanta, MTU Min Saturation).
RESERVED	31:17	0x0	RSV	N/A	Reserved.

13.2.2.17.59 DCB Transmit Wait Bulk DWRR Saturation Value - GLTCB_BULK_DWRR_WB_SAT (0x000AE0F4; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SATURATION	16:0	0x2000	RW	UNDEFINED	**Saturation** Bulk wait buffer saturation value in bytes for all ports. Credits cannot be added over this value. Minimal Saturation value depends on Port MTU Size (for all quantas): MTU is [0k-2k]: 3K MTU is [2k-4k]: 6K MTU is [4k-8k]: 12K MTU is [8k-16k] (Jumbo): 24K Final Minimal Saturation = Max(Quanta, MTU Min Saturation).

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	31:17	0x0	RSV	N/A	Reserved.

13.2.2.17.60 DCB Transmit Regular Low Latency DWRR Saturation Value - GLTCB_LL_DWRR_REG_SAT (0x000AE0F8; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SATURATION	16:0	0x2000	RW	UNDEFINED	**Saturation** Low latency regular buffer saturation value in bytes for all ports. Credits cannot be added over this value. Minimal Saturation value depends on Port MTU Size (for all quantas): MTU is [0k-2k]: 3K MTU is [2k-4k]: 6K MTU is [4k-8k]: 12K MTU is [8k-16k] (Jumbo): 24K Final Minimal Saturation = Max(Quanta, MTU Min Saturation).
RESERVED	31:17	0x0	RSV	N/A	Reserved.

13.2.2.17.61 DCB Transmit Wait Low Latency DWRR Saturation Value - GLTCB_LL_DWRR_WB_SAT (0x000AE0FC; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SATURATION	16:0	0x2000	RW	UNDEFINED	**Saturation** Low latency wait buffer saturation value in bytes for all ports. Credits cannot be added over this value. Minimal Saturation value depends on Port MTU Size (for all quantas): MTU is [0k-2k]: 3K MTU is [2k-4k]: 6K MTU is [4k-8k]: 12K MTU is [8k-16k] (Jumbo): 24K Final Minimal Saturation = Max(Quanta, MTU Min Saturation).
RESERVED	31:17	0x0	RSV	N/A	Reserved.

13.2.2.17.62 TCB Arbiter Credit Expansion - PRTTCB_CREDIT_EXP (0x000AE100; RW)

Register used to add arbiter credit expansion for packets per port.

Field	Bit(s)	Init.	Type	CFG Policy	Description
EXPANSION	7:0	0x18	RW	UNDEFINED	**Expansion** Amount of credits in bytes to be added when updating the arbiter. Can be used to allow arbiter to take into account MAC IPG and so on.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.17.63 TCB Arbiter Credit Expansion Control - GLTCB_CREDIT_EXP_CTL (0x000AE120; RW)

Used to enable/disable arbiter credit expansion and set minimal packet size for updating credits.

Field	Bit(s)	Init.	Type	CFG Policy	Description
EN	0	0b	RW	UNDEFINED	**Enable** Enables credit expansion. 0b = Packet length received is updated to arbiter. 1b = Packet length is expanded according to: new_len = Max(orig_len + expansion, min_pkt)
MIN_PKT	9:1	0x80	RW	UNDEFINED	**Minimal Packet** Used to enforce minimal size for arbiter credit update, expressed in bytes. This value is updated to arbiter for packets smaller than (min_pkt - configured expansion), otherwise the packet size + configured expansion is updated.
RESERVED	31:10	0x0	RSV	N/A	Reserved.

13.2.2.17.64 Global MNG LL Strict Priority Enable - GLDCB_TCB_MNG_SP (0x000AE12C; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MNG_SP	0	1b	RW	UNDEFINED	**MNG Strict Priority** When set, low latency MNG packets get strict priority over low latency wait/regular buffer packets of the same port in TCB arbitration. Otherwise, the wait/regular buffer packets get strict priority over the MNG packets.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.17.65 TC Low Latency Config - GLDCB_TCB_TCLL_CFG (0x000AE134; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LLTC	31:0	0x0	RW	UNDEFINED	**Low Latency TC** Bit per each TC. Each entry controls whether the TC is considered to have low latency needs for the transmit path. 0b = TC is defined to be a Bulk. 1b = TC is defined to be a Low Latency TC for transmit.

13.2.2.17.66 DCB Transmit Rate Limiter Control per TC - TCTCB_WB_RL_TC_CFG[n] (0x000AE138 + 0x4*n, n=0...31; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TOKENS	11:0	0x381	RW	UNDEFINED	**Tokens** Amount of tokens in bytes added to bucket after each period (period is configured) per TC.
BURST_SIZE	21:12	0x017	RW	UNDEFINED	**Burst Size** Max amount of tokens a bucket can hold in 64-byte resolutions (up to 64KB) per TC.
RESERVED	31:22	0x0	RSV	N/A	Reserved.

13.2.2.17.67 DCB Transmit Rate Limiter Status per TC - TCTCB_WB_RL_TC_STAT[n] (0x000AE1B8 + 0x4*n, n=0...31; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BUCKET	16:0	0x0	RO	N/A	**Bucket** Current amount of tokens in bytes that are in the bucket per TC.Coded in 2's compliment.
RESERVED	31:17	0x0	RSV	N/A	Reserved.

13.2.2.17.68 TC Rate Limiters Config - GLTCB_WB_RL (0x000AE238; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PERIOD	15:0	0x0020	RW	UNDEFINED	**Period** Amount of clock cycles to wait before adding new tokens to bucket. Every PERIOD CC, TCB_WB_RL_TC_CFG.*TOKENS* bytes are added to TCB_WB_RL_TC_STAT.*BUCKET*.
EN	16	0b	RW	UNDEFINED	**Enable** 0b = Disables TCB wait buffer rate limiters. They no longer block WB traffic and receive credit updates. 1b = Enables TCB wait buffer rate limiters.
RESERVED	31:17	0x0	RSV	N/A	Reserved.

13.2.2.17.69 Global Wait Buffer Strict Priority Enable - GLDCB_TCB_WB_SP (0x000AE310; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
WB_SP	0	0b	RW	UNDEFINED	**Wait Buffer Strict Priority** When set, wait buffer packets get strict priority over regular buffer packets of the same port in the same LL/Bulk class, in TCB arbitration. Otherwise, the wait/regular buffer arbitration is done using the weighted round-robin algorithm.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.17.70 DCB Transmit Command Pipe Port Monitor Status - PRTDCB_TCUPM_REG_CM (0x000BC360; RO)

Regular buffer command monitor status per port.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MONITOR	14:0	0x0	RW	UNDEFINED	**Monitor** Regular buffer port commands monitor status. Current amount of commands accounted by the Per Port Tx LAN Command Pipe Monitor.
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.17.71 DCB Transmit Command Pipe Port Monitor Threshold - PRTDCB_TCUPM_REG_CTHR (0x000BC380; RW)

Regular buffer command monitor threshold per port.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PORTOFFTH_H	14:0	0x43C	RW	UNDEFINED	**Port OFF Threshold High** Regular buffer port OFF commands high threshold. Depth of the per Port Monitor applied over the Tx LAN/RDMA Command Pipes altogether. It is expressed in commands units. Blocks port when monitor is over threshold.
PORTOFFTH_L	29:15	0x43C	RW	UNDEFINED	**Port OFF Threshold Low** regular buffer port OFF commands low threshold. Depth of the per Port Monitor applied over the Tx LAN/RDMA Command Pipes altogether. It is expressed in commands units. Blocks only Legacy TCs of the port when monitor is over threshold.
RESERVED	31:30	0x0	RSV	N/A	Reserved.

13.2.2.17.72 DCB Transmit Data Pipe Port Monitor Status - PRTDCB_TCUPM_REG_DM (0x000BC3A0; RO)

Regular buffer data monitor status per port.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MONITOR	18:0	0x0	RW	UNDEFINED	**Monitor** Regular buffer port data monitor status. Current amount of bytes accounted by the Per Port Tx LAN Data Pipe Monitor.
RESERVED	31:19	0x0	RSV	N/A	Reserved.

13.2.2.17.73 DCB Transmit Data non-Exceed Pipe Monitor Status - PRTDCB_TCUPM_NO_EXCEED_DM (0x000BC3C0; RO)

Regular buffer non-exceed data monitor status.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MONITOR	18:0	0x0	RW	UNDEFINED	**Monitor** Regular buffer non-exceed data monitor status. Total number of bytes in the TCB waiting and regular buffer for all PFC-enabled TCs of the port. Counts scheduled data only (not including exceeded quantas).
RESERVED	31:19	0x0	RSV	N/A	Reserved.

13.2.2.17.74 DCB Transmit Data Pipe Port Monitor Threshold - PRTDCB_TCUPM_REG_DTHR (0x000BC3E0; RW)

Regular buffer data monitor threshold per port.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PORTOFFTH_H	11:0	0x98	RW	UNDEFINED	**Port OFF Threshold High** Regular buffer port OFF data high threshold. Depth of the per Port Monitor applied over the Tx LAN/RDMA Command Pipes altogether. It is expressed in 128-byte units. Threshold is checked against the non-Exceed Data Monitor. When GLDCB_TCUPM_GENC.*NON_EXCEED_DIS* is high, threshold applies to the Regular Data Monitor instead. Blocks port when monitor is over threshold.
PORTOFFTH_L	23:12	0x98	RW	UNDEFINED	**Port OFF Threshold Low** Regular buffer port OFF data low threshold. Depth of the per Port Monitor applied over the Tx LAN/RDMA Command Pipes altogether. It is expressed in 128-byte units. Blocks only Legacy TCs of the port when monitor is over threshold.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.17.75 DCB Transmit Data Pipe Port Monitor Status - PRTDCB_TCUPM_REG_PE_HB_DM (0x000BC400; RO)

Regular buffer PE headers monitor status per port.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MONITOR	11:0	0x0	RW	UNDEFINED	**Monitor** Regular buffer port PE header buffers monitor status. Current amount of bytes accounted by the per port PE Header Buffers Pipe Monitor, in 32byte units.
RESERVED	31:12	0x0	RSV	N/A	Reserved.

13.2.2.17.76 DCB Transmit Data Pipe Port Monitor Threshold - PRTDCB_TCUPM_REG_PE_HB_DTHR (0x000BC420; RW)

Regular buffer PE headers monitor threshold per port.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PORTOFFTH_H	11:0	0x98	RW	UNDEFINED	**Port OFF Threshold High** Regular buffer port OFF PE header buffers high threshold. It is expressed in 32-byte units. Blocks port when monitor is over threshold.
PORTOFFTH_L	23:12	0x98	RW	UNDEFINED	**Port OFF Threshold Low** Regular buffer port OFF PE header buffers low threshold. It is expressed in 32-byte units. Blocks only Legacy TCs of the port when monitor is over threshold.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.17.77 DCB Transmit Command Pipe Port Waiting Monitor Status - PRTDCB_TCUPM_WAIT_PFC_CM (0x000BC440; RO)

Wait + regular buffer command monitor status per port of PFC-enabled TCs only.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MONITOR	14:0	0x0	RW	UNDEFINED	**Monitor** Wait buffer port commands monitor status. Total number of commands in the TCB waiting and regular buffer for all PFC-enabled TCs of the port.
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.17.78 DCB Transmit Command Pipe Port Waiting Monitor Threshold - PRTDCB_TCUPM_WAIT_PFC_CTHR (0x000BC460; RW)

Wait + regular buffer command monitor threshold per port.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PORTOFFTH	14:0	0x9F8	RW	UNDEFINED	**Port OFF Threshold** Wait buffer port OFF commands threshold. High threshold on the total amount of commands that is waiting in the TCB waiting lists of the port for all its PFC-enabled TCs. Only PFC-enabled TCs of the port are flow controlled when this threshold is crossed.
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.17.79 DCB Transmit Data Pipe Port Waiting Monitor Status - PRTDCB_TCUPM_WAIT_PFC_DM (0x000BC480; RO)

Wait + regular buffer data monitor status per port of PFC-enabled TCs only.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MONITOR	18:0	0x0	RW	UNDEFINED	**Monitor** Wait buffer port data monitor status. Total number of bytes in the TCB waiting and regular buffer for all PFC-enabled TCs of the port.
RESERVED	31:19	0x0	RSV	N/A	Reserved.

13.2.2.17.80 DCB Transmit Data Pipe Port Waiting Monitor Threshold - PRTDCB_TCUPM_WAIT_PFC_DTHR (0x000BC4A0; RW)

Wait + regular buffer data monitor threshold per port.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PORTOFFTH	11:0	0x274	RW	UNDEFINED	**Port OFF Threshold** Wait buffer port OFF data threshold. High threshold on the total amount of bytes in 128-byte units that is waiting in the TCB waiting lists of the port for all its PFC-enabled TCs. Only PFC-enabled TCs of the port are flow controlled when this threshold is crossed.
RESERVED	31:12	0x0	RSV	N/A	Reserved.

13.2.2.17.81 DCB Transmit Data Pipe Port Waiting Monitor Status - PRTDCB_TCUPM_WAIT_PFC_PE_HB_DM (0x000BC4C0; RO)

Wait + regular buffer PE headers monitor status per port of PFC enabled TCs only.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MONITOR	11:0	0x0	RW	UNDEFINED	**Monitor** Wait buffer port PE header buffers monitor status. Expressed in 32-byte units. Total number of PE headers bytes in the TCB waiting and regular buffer for all PFC enabled TCs of the port.
RESERVED	31:12	0x0	RSV	N/A	Reserved.

13.2.2.17.82 DCB Transmit Data Pipe Port Waiting Monitor Threshold - PRTDCB_TCUPM_WAIT_PFC_PE_HB_DTHR (0x000BC4E0; RW)

Wait + regular buffer PE headers monitor threshold per port.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PORTOFFTH	11:0	0x130	RW	UNDEFINED	**Port OFF Threshold** Wait buffer port OFF PE header buffers threshold. High threshold on the total amount of bytes in 32-byte units that is waiting in the TCB waiting lists of the port for all its PFC-enabled TCs. Only PFC-enabled TCs of the port are flow controlled when this threshold is crossed.
RESERVED	31:12	0x0	RSV	N/A	Reserved.

13.2.2.17.83 DCB Transmit Command Pipe TC Waiting Monitor Status - TCDCB_TCUPM_WAIT_CM[n] (0x000BC520 + 0x4*n, n=0...31; RO)

Wait + regular buffer command monitor status per TC.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MONITOR	14:0	0x0	RW	UNDEFINED	**Monitor** Wait buffer TC commands monitor status. Number of commands that are in transit from the host to the TCB (TCB waiting list included) for TC n, where n is the index of the register in the array.
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.17.84 DCB Transmit Command Pipe TC Waiting Monitor Threshold - TCDCB_TCUPM_WAIT_CTHR[n] (0x000BC5A0 + 0x4*n, n=0...31; RW)

Wait + regular buffer command monitor threshold per TC.

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCOFFTH	14:0	0x4BC	RW	UNDEFINED	**TC OFF Threshold** Wait buffer TC OFF commands threshold. The total amount of commands that can accumulate in the TCB (waiting list included) of the port for the same PFC-enabled TC.
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.17.85 DCB Transmit Data Pipe TC Waiting Monitor Status - TCDCB_TCUPM_WAIT_DM[n] (0x000BC620 + 0x4*n, n=0...31; RO)

Wait + regular buffer data monitor status per TC.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MONITOR	18:0	0x0	RW	UNDEFINED	**Monitor** Wait Buffer TC data monitor status. Number of bytes that are in transit from the host to the TCB (TCB waiting list included) for TC n, where n is the index of the register in the array.
RESERVED	31:19	0x0	RSV	N/A	Reserved.

13.2.2.17.86 DCB Transmit Data Pipe TC Waiting Monitor Threshold - TCDCB_TCUPM_WAIT_DTHR[n] (0x000BC6A0 + 0x4*n, n=0...31; RW)

Wait + regular buffer data monitor threshold per TC.

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCOFFTH	11:0	0x104	RW	UNDEFINED	**TC OFF Threshold** Wait buffer TC OFF Data Threshold. The total amount of bytes in 128-byte units that can accumulate in the TCB (waiting list included) of the port for the same PFC-enabled TC.
RESERVED	31:12	0x0	RSV	N/A	Reserved.

13.2.2.17.87 DCB Transmit Data Pipe TC Waiting Monitor Status - TCDCB_TCUPM_WAIT_PE_HB_DM[n] (0x000BC720 + 0x4*n, n=0...31; RO)

Wait + regular buffer PE headers monitor status per TC.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MONITOR	11:0	0x0	RW	UNDEFINED	**Monitor** Wait buffer TC PE header buffers monitor status. Expressed in 32-byte units. Number of bytes that are in transit from the host to the TCB (TCB waiting list included) for TC n, where n is the index of the register in the array.
RESERVED	31:12	0x0	RSV	N/A	Reserved.

13.2.2.17.88 DCB Transmit Data Pipe TC Waiting Monitor Threshold - TCDCB_TCUPM_WAIT_PE_HB_DTHR[n] (0x000BC7A0 + 0x4*n, n=0...31; RW)

Wait + regular buffer PE headers monitor threshold per TC.

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCOFFTH	11:0	0x98	RW	UNDEFINED	**TC OFF Threshold** Wait buffer TC OFF PE header buffers threshold. The total amount of bytes in 32-byte units that can accumulate in the TCB (waiting list included) of the port for the same PFC-enabled TC.
RESERVED	31:12	0x0	RSV	N/A	Reserved.

13.2.2.17.89 DCB TC Immediate FC Mode - GLDCB_TCUPM_IMM_EN (0x000BC824; RW)

Configures FC immediate mode for each TC.

Field	Bit(s)	Init.	Type	CFG Policy	Description
IMM_EN	31:0	0xFFFFFFFF	RW	UNDEFINED	**Immediate Enable** Per TC Tx-Pipe immediate FC mode. 0b = Conditioned XOFF Forwarding Mode (default). XOFF notifications received from the TC data pipe monitor are internally forwarded upward to the Tx-Scheduler only once the command pipe monitor for this TC is full. This mode provides better XON recovering time. 1b = Immediate XOFF Forwarding Mode. XOFF notifications received from the TC data pipe monitor are immediately forwarded internally upward to the Tx-Scheduler. This mode is useful to support PFC-enabled TCs of the port like true independent traffic classes when more than two such TCs are configured over the port. This bit should be kept as zero.

13.2.2.17.90 DCB TC Legacy Queues Mapping - GLDCB_TCUPM_LEGACY_TC (0x000BC828; RW)

Configure which TCs send legacy queues.

Field	Bit(s)	Init.	Type	CFG Policy	Description
LEGTC	31:0	0x0	RW	UNDEFINED	**Legacy TC** Per TC configuration to indicate if it is sending legacy queues. Only legacy TCs are halted by the regular buffer data monitor (instead of entire port) when over low threshold. The port is blocked by the non-exceed monitor when over high threshold.

13.2.2.17.91 DCB Transmit Data non-Exceed Monitor Enable - GLDCB_TCUPM_NO_EXCEED_DIS (0x000BC830; RW)

TCUPM non-exceed regular buffer data monitor enable.

Field	Bit(s)	Init.	Type	CFG Policy	Description
NON_EXCEED_DIS	0	1b	RW	UNDEFINED	**Non-Exceed Disable** Disables non-exceed monitor. When disabled, the regular buffer high threshold is applied over the regular (exceed) monitor.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.17.92 DCB Transmit Wait Port Data Monitor Enable - GLDCB_TCUPM_WB_DIS (0x000BC834; RW)

TCUPM wait buffer port/TC data monitor disable.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PORT_DISABLE	0	1b	RW	UNDEFINED	**Port Disable** Disables wait buffer port data monitors.
TC_DISABLE	1	0b	RW	UNDEFINED	**TC Disable** Disables wait buffer TC data monitors.
RESERVED	31:2	0x0	RSV	N/A	Reserved.

13.2.2.17.93 DCB Receive Port Round Robin Control - PRTDCB_RPRRC (0x001220C0; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BWSHARE	9:0	0x14	RW	UNDEFINED	**Bandwidth Share** Relative weight of the available bandwidth allocated to the port in Rx. The value of 1 is used for ports operated at 100 Mb/s speed.
RESERVED	30:10	0x0	RSV	N/A	Reserved.
BWSHARE_DIS	31	1b	RW	UNDEFINED	**Bandwidth Share Disable** Disable weighted arbitration. Disabled ports arbitrate in packet-based round-robin manner.

13.2.2.17.94 DCB Receive Port Round Robin Status - PRTDCB_RPRRS (0x001220E0; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CREDITS	31:0	0x0	RW	UNDEFINED	**Credits** Current amount of credits accumulated by the port in Rx. The amount is expressed in byte units and it is formatted as an algebraic 2's complement number. Writing to this field has the effect of loading a new current credit value, which is used by Rx-PRR algorithm.

13.2.2.17.95 QRX - GLDCB_RTC2PFC_RCB (0x00122100; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TC2PFC	31:0	0x0	RW	UNDEFINED	**TC-to-Priority Flow Control** Bitmap that controls the use of Priority Flow Control (PFC) per each TC. Bit n set to 1b: TC n uses PFC in Rx and Tx. The TC is referred as a no-drop TC. Bit n clear to 0b: The device does not issue PFC pause frames with bits set to 1b in the *priority_enable_vector* for the UPs attached to that TC. It does not react to bits set to 1b for the UPs attached to that TC in the priority_enable_vector of a received PFC pause frame. The TC is referred as a drop UP. For up to four link topology: Bits 0-7 are for port 0, TC 0 to 7. Bits 8-15 are for port 1, TC 0 to 7. Bits 16-23 are for port 2, TC 0 to 7. Bits 24-31 are for port 3, TC 0 to 7. For more than four link topology: Bits 00-03 are for port 0, TC 0 to 3. Bits 04-07 are for port 1, TC 0 to 3. Bits 08-11 are for port 2, TC 0 to 3. Bits 12-15 are for port 3, TC 0 to 3. Bits 16-19 are for port 4, TC 0 to 3. Bits 20-23 are for port 5, TC 0 to 3. Bits 24-27 are for port 6, TC 0 to 3. Bits 28-31 are for port 7, TC 0 to 3.

13.2.2.17.96 DCB Receive ETS per TC Control - GLDCB_RETSTCC[n] (0x00122140 + 0x4*n, n=0...31; RW)

One register per TC. Register index corresponds to TCID.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	30:0	0x0	RSV	N/A	Reserved.
ETSTC	31	0b	RW	UNDEFINED	**ETS TC** Controls the use of ETS as the Transmit Selection Algorithm (TSA) in Rx for TC n, where n is the register index in the array. 0b = TC n uses a Strict Priority or other TSA in Rx. 1b = TC n uses ETS scheme in Rx.

13.2.2.17.97 DCB Receive ETS per TC Status - GLDCB_RETSTCS[n] (0x001221C0 + 0x4*n, n=0...31; RW)

One register per TC. Register index corresponds to TCID.

Field	Bit(s)	Init.	Type	CFG Policy	Description
CREDITS	31:0	0x0	RW	UNDEFINED	**Credits** Current amount of credits accumulated by TC n in Rx, where n is the register index in the array. The amount is expressed in byte units and it is formatted as an algebraic 2's complement number. Writing to this field has the effect of loading a new current credit value, which is used by Rx-ETS algorithm.

13.2.2.17.98 DCB Receive ETS Control - PRTDCB_RETSC (0x001222A0; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	0	1b	RSV	N/A	Reserved.
NON_ETS_MODE	1	1b	RW	UNDEFINED	**Non-ETS Mode** Rx non-ETS operating mode. 0b = Strict Priority (SP) mode. 1b = Round Robin (RR) mode.
RESERVED	31:2	0x0	RSV	N/A	Reserved.

13.2.2.17.99 DCB Receive UP to TC Mapping - PRTDCB_RUP2TC (0x001D2640; RW)

Bitmap long by 24 bits, with a 3-bit entry per each UP. Each entry controls the mapping of a UP to a 3-bit TC index in receive. Higher TC index means higher priority of the traffic class.

The same mapping is used for packets received from the wires and for those looped back internally. It defines on the account of which TC a packet is stored in the Rx packet buffer, and which UPs bits are set in the PFC XOFF/XON frames issued to the link partner when the filling state of a TC requires it.

Default mapping maps all UPs to TC0.

Field	Bit(s)	Init.	Type	CFG Policy	Description
UP0TC	2:0	000b	RW	UNDEFINED	**UP 0 TC** TC index to which UP 0 is mapped.
UP1TC	5:3	000b	RW	UNDEFINED	**UP 1 TC** TC index to which UP 1 is mapped.
UP2TC	8:6	000b	RW	UNDEFINED	**UP 2 TC** TC index to which UP 2 is mapped.
UP3TC	11:9	000b	RW	UNDEFINED	**UP 3 TC** TC index to which UP 3 is mapped.
UP4TC	14:12	000b	RW	UNDEFINED	**UP 4 TC** TC index to which UP 4 is mapped.
UP5TC	17:15	000b	RW	UNDEFINED	**UP 5 TC** TC index to which UP 5 is mapped.
UP6TC	20:18	000b	RW	UNDEFINED	**UP 6 TC** TC index to which UP 6 is mapped.
UP7TC	23:21	000b	RW	UNDEFINED	**UP 7 TC** TC index to which UP 7 is mapped.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.17.100 DCB TC to PFC Mapping - GLDCB_TC2PFC (0x001D2694; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TC2PFC	31:0	0x0	RW	UNDEFINED	**TC-to-Priority Flow Control** Bitmap that controls the use of Priority Flow Control (PFC) per each TC. Bit n set to 1b: TC n uses PFC in Rx and Tx. The TC is referred as a no-drop TC. Bit n clear to 0b: The device does not issue PFC pause frames with bits set to 1b in the *priority_enable_vector* for the UPs attached to that TC. It does not react to bits set to 1b for the UPs attached to that TC in the *priority_enable_vector* of a received PFC pause frame. The TC is referred as a drop UP.

13.2.2.17.101 DCB Transmit UP to TC Mapping - PRTDCB_TUP2TC (0x001D26C0; RW)

Bitmap long by 24 bits, with a 3-bit entry per each UP. Each entry controls the mapping of a UP to a 3-bit TC index in receive. Higher TC index means higher priority of the traffic class.

The same mapping is used for packets received from the wires and for those looped back internally. It defines on the account of which TC a packet is stored in the Rx packet buffer, and which UPs bits are set in the PFC XOFF/XON frames issued to the link partner when the filling state of a TC requires it.

Default mapping maps all UPs to TC0.

Field	Bit(s)	Init.	Type	CFG Policy	Description
UP0TC	2:0	000b	RW	UNDEFINED	**UP 0 TC** TC index to which UP 0 is mapped.
UP1TC	5:3	000b	RW	UNDEFINED	**UP 1 TC** TC index to which UP 1 is mapped.
UP2TC	8:6	000b	RW	UNDEFINED	**UP 2 TC** TC index to which UP 2 is mapped.
UP3TC	11:9	000b	RW	UNDEFINED	**UP 3 TC** TC index to which UP 3 is mapped.
UP4TC	14:12	000b	RW	UNDEFINED	**UP 4 TC** TC index to which UP 4 is mapped.
UP5TC	17:15	000b	RW	UNDEFINED	**UP 5 TC** TC index to which UP 5 is mapped.
UP6TC	20:18	000b	RW	UNDEFINED	**UP 6 TC** TC index to which UP 6 is mapped.
UP7TC	23:21	000b	RW	UNDEFINED	**UP 7 TC** TC index to which UP 7 is mapped.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.17.102 Transmit Flow Control Status - PRTDCB_TFCS (0x001E4560; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TXOFF	0	0b	RO	N/A	**Tx Off** Transmission Paused. Pause state indication of the transmit function when symmetrical link flow control is enabled.
RESERVED	7:1	0x0	RSV	N/A	Reserved.
TXOFF0	8	0b	RO	N/A	**Tx Off 0** TC 0 Transmission Paused. Pause state indication of the TC 0 when priority flow control is enabled.
TXOFF1	9	0b	RO	N/A	**Tx Off 1** TC 1 Transmission Paused. Pause state indication of the TC 1 when priority flow control is enabled.
TXOFF2	10	0b	RO	N/A	**Tx Off 2** TC 2 Transmission Paused. Pause state indication of the TC 2 when priority flow control is enabled.
TXOFF3	11	0b	RO	N/A	**Tx Off 3** TC 3 Transmission Paused. Pause state indication of the TC 3 when priority flow control is enabled.
TXOFF4	12	0b	RO	N/A	**Tx Off 4** TC 4 Transmission Paused. Pause state indication of the TC 4 when priority flow control is enabled.
TXOFF5	13	0b	RO	N/A	**Tx Off 5** TC 5 Transmission Paused. Pause state indication of the TC 5 when priority flow control is enabled.
TXOFF6	14	0b	RO	N/A	**Tx Off 6** TC 6 Transmission Paused. Pause state indication of the TC 6 when priority flow control is enabled.
TXOFF7	15	0b	RO	N/A	**Tx Off 7** TC 7 Transmission Paused. Pause state indication of the TC 7 when priority flow control is enabled.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.17.103 Flow Control Transmit Timer Value n - PRTDCB_FCTTVN[n] (0x001E4580 + 0x20*n, n=0...3; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TTV_2N	15:0	0xFFFF	RW	UNDEFINED	**Transmit Timer Value 2n** Timer value included in XOFF frames as Timer (2n). The same value must be set to User Priorities attached to the same TC, as defined in PRTDCB_RUP2TC register. For legacy 802.3x flow control packets, TTV0 is the only timer that is used.
TTV_2N_P1	31:16	0xFFFF	RW	UNDEFINED	**Transmit Timer Value 2n+1** Timer value included in XOFF frames as Timer (2n+1). The same value must be set to User Priorities attached to the same TC, as defined in PRTDCB_RUP2TC register.

13.2.2.17.104 Flow Control Refresh Threshold Value - PRTDCB_FCRTV (0x001E4600; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FC_REFRESH_TH	15:0	0x7FFF	RW	UNDEFINED	**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 user priority N is: FCTTV[N/2].*TTV*[Nmod2] - FCRTV.*FC_REFRESH_TH*
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.17.105 Flow Control Configuration - PRTDCB_FCCFG (0x001E4640; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	2:0	000b	RSV	N/A	Reserved.
TFCE	4:3	00b	RW	UNDEFINED	**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.
RESERVED	31:5	0x0	RSV	N/A	Reserved.

13.2.2.17.106 DCB Transmit PFC Timer Status - PRTDCB_TPFCTS[n] (0x001E4660 + 0x20*n, n=0...7; RW)

One register per TC. Register index corresponds to TCID.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PFCTIMER	13:0	0x0	RW	UNDEFINED	**Priority Flow Control Timer** Current value of the PFC Timer of TC n in Tx, where n is the register index in the array. The amount is expressed in milliseconds. The timer saturates to 0x3FFF. Writing to this field has the effect of loading a new current timer value, which can be useful for diagnostic purposes.
RESERVED	31:14	0x0	RSV	N/A	Reserved.

13.2.2.17.107 DCB Receive ETS per TC Control - GLDCB_PRS_RETSTCC[n] (0x002000B0 + 0x4*n, n=0...31; RW)

One register per TC. Register index corresponds to TCID.

Field definitions are the same as those defined in Section 13.2.2.17.96.

13.2.2.17.108 DCB Receive Shared Pipe Monitor Control - GLDCB_PRS_RSPMC (0x00200160; RW)

Field definitions are the same as those defined in Section 13.2.2.31.8.

13.2.2.17.109 DCB Receive Port Round Robin Control - PRTDCB_PRS_RPRRC (0x00200180; RW)

Field definitions are the same as those defined in Section 13.2.2.17.93.

13.2.2.17.110 DCB Receive ETS Control - PRTDCB_PRS_RETSC (0x002001A0; RW)

Field definitions are the same as those defined in Section 13.2.2.17.98.

13.2.2.17.111 DCB Receive ETS per TC Control - GLDCB_SWT_RETSTCC[n] (0x0020A040 + 0x4*n, n=0...31; RW)

One register per TC. Register index corresponds to TCID.

Field definitions are the same as those defined in Section 13.2.2.17.96.

13.2.2.17.112 DCB Receive ETS Control - PRTDCB_SWT_RETSC (0x0020A140; RW)

Field definitions are the same as those defined in Section 13.2.2.17.98.

13.2.2.18 PF - Receive Packet Buffer Registers

13.2.2.18.1 RPB Dedicated Pool High Watermark - GLRPB_DHW[n] (0x000AC000 + 0x4*n, n=0...15; RW)

One register per TC. Register index corresponds to TCID.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DHW_TCN	19:0	0x0	RW	UNDEFINED	**Dedicated Pool High Watermark** It is expressed in bytes.
RESERVED	31:20	0x0	RSV	N/A	Reserved.

13.2.2.18.2 RPB Dedicated Pool Low Watermark - GLRPB_DLW[n] (0x000AC044 + 0x4*n, n=0...15; RW)

One register per TC. Register index corresponds to TCID.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DLW_TCN	19:0	0x0	RW	UNDEFINED	**Dedicated Pool Low Watermark** It is expressed in bytes.
RESERVED	31:20	0x0	RSV	N/A	Reserved.

13.2.2.18.3 RPB Dedicated Pool Size - GLRPB_DPS[n] (0x000AC084 + 0x4*n, n=0...15; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DPS_TCN	19:0	0x0	RW	UNDEFINED	**Dedicated Pool Size** It is expressed in bytes.
RESERVED	31:20	0x0	RSV	N/A	Reserved.

13.2.2.18.4 RPB Shared Pool Size - GLRPB_SPS[n] (0x000AC0C4 + 0x4*n, n=0...7; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SPS_TCN	19:0	0xE1000	RW	UNDEFINED	**Shared Pool Size** It is expressed in bytes.
RESERVED	31:20	0x0	RSV	N/A	Reserved.

13.2.2.18.5 RPB Shared Pool High Watermark - GLRPB_SHW[n] (0x000AC120 + 0x4*n, n=0...7; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SHW	19:0	0xC8000	RW	UNDEFINED	**Shared Pool High Watermark** It is expressed in bytes.
RESERVED	31:20	0x0	RSV	N/A	Reserved.

13.2.2.18.6 RPB Shared Pool Low Watermark - GLRPB_SLW[n] (0x000AC140 + 0x4*n, n=0...7; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SLW	19:0	0x0	RW	UNDEFINED	**Shared Pool Low Watermark** It is expressed in bytes.
RESERVED	31:20	0x0	RSV	N/A	Reserved.

13.2.2.18.7 TC Pool Config - GLRPB_TC_CFG[n] (0x000AC2A4 + 0x4*n, n=0...31; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
D_POOL	15:0	0x8	RW	UNDEFINED	**Dedicated Pool** TC dedicated pool index. Allowed values are 0..7.
S_POOL	31:16	0x1	RW	UNDEFINED	**Shared Pool** TC shared pool index. Allowed values are 0..15.

13.2.2.18.8 DSI Traffic Enable - GLRPB_DSI_EN (0x000AC324; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DSI_EN	0	0b	RW	UNDEFINED	**DSI Enable**
DSI_L2_MAC_ERR_DROP_EN	1	1b	RW	UNDEFINED	**DSI L2 MAC Error Drop Enable** Enables the dropping of DSI packets that have l2_mac_error (CRC, oversize, and so on).
RESERVED	31:2	0x0	RSV	N/A	Reserved.

13.2.2.18.9 RPB TC High Watermark - GLRPB_TCHW[n] (0x000AC330 + 0x4*n, n=0...31; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCHW	19:0	0x19000	RW	UNDEFINED	**TC High Watermark** It is expressed in bytes.
RESERVED	31:20	0x0	RSV	N/A	Reserved.

13.2.2.18.10 RPB TC Low Watermark - GLRPB_TCLW[n] (0x000AC3B0 + 0x4*n, n=0...31; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCLW	19:0	0x0	RW	UNDEFINED	**TC Low Watermark** It is expressed in bytes.
RESERVED	31:20	0x0	RSV	N/A	Reserved.

13.2.2.19 PF - Transmit Scheduler Registers

Registers related to the Transmit Scheduler.

13.2.2.19.1 TSCD PEPM - GLPE_TSCD_PEPM (0x0051E228; RO)

Credit information for TSCD requests from PEPM.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MDQ_CREDITS	7:0	0x29	RW	UNDEFINED	**MDQ Credits** The number of MDQ credits TSCD requests from PEPM.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.19.2 Transmit Scheduler FLR - GLPE_TSCD_FLR[n] (0x0051E24C + 0x4*n, n=0...3; RO)

CQP writes these registers to trigger FLR flow through PE.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DRAIN_VCTR_ID	1:0	00b	RW	UNDEFINED	**Drain Vector ID**
PORT	4:2	000b	RW	UNDEFINED	**Port**
PF_NUM	7:5	000b	RW	UNDEFINED	**PF Number**
VM_VF_TYPE	9:8	00b	RW	UNDEFINED	**VM/VF Type**
RESERVED	15:10	0x0	RSV	N/A	Reserved.
VM_VF_NUM	25:16	0x0	RW	UNDEFINED	**VM/VF Number**
RESERVED	30:26	0x0	RSV	N/A	Reserved.
VLD	31	0b	RW	UNDEFINED	**Valid**

13.2.2.19.3 Transmit Scheduler Number of PQS - GLPE_TSCD_NUM_PQS (0x0051E2FC; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
NUM_PQS	31:0	0x120	RO	N/A	**Number of PQs** The number of PQs that the Transmit Scheduler was built with.

13.2.2.20 PF - Host Memory Cache Registers

Registers for Host Memory Cache.

13.2.2.20.1 FOC Cache Attributes - GLFOC_CACHESIZE (0x000AA074; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
WORD_SIZE	7:0	0x40	RO	N/A	**Word Size** The cache line size in bytes.
SETS	19:8	0x80	RO	N/A	**Sets** The number of cache sets.
WAYS	23:20	0x8	RO	N/A	**Ways** The number of cache ways.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.20.2 Private Memory Space Segment Descriptor Command - PFHMC_SDCMD_FPMAT (0x00100000; RW)

This register is used to access the Host Memory Cache's segment table. The HMC's segment table is partitioned per Private Memory Function (PMF), and this register is only allowed to access the Segment Table entries allocated to the PMF associated with this register via the SD Partition register, GLHMC_SDPART.

For read operations, PFHMC_SDCMD must be written with *PMSDWR* set to 0b and *PMSDIDX* set to the segment table index to be read. After the write to PFHMC_SDCMD completes, PFHMC_SDDATALOW and PFHMC_SDDATAHIGH can be read to retrieve the segment descriptor contents.

For write operations, PFHMC_SDDATALOW and PFHMC_SDDATAHIGH must be written before writing PFHMC_SDCMD with *PMSDWR* set to 1b and *PMSDIDX* set to the segment table index to be written.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMSDIDX	11:0	0x0	RW	UNDEFINED	**Private Memory Segment Descriptor Index** Relative Index of the HMC Segment Descriptor to be read or written. The actual absolute index to be used to access the Segment Table is (*PMSDBASE* + *PMSDIDX*), where *PMSDBASE* is from the *PMSDBASE* field of the GLHMC_SDPART or GLHMC_PFPESDPART register associated with this function. On write operations, if (*PMSDIDX* >= *PMSDSIZE*), the write is dropped, where *PMSDSIZE* is from the *PMSDSIZE* field of the GLHMC_SDPART or GLHMC_PFPESDPART register associated with this function. The *PMSDPARTSEL* bit of this register determines which SD Partition register is selected for the absolute SD index calculation and for the check that *PMSDIDX* is within the range allowed by this function.
RESERVED	14:12	000b	RSV	N/A	Reserved.
PMSDPARTSEL	15	0b	RW	UNDEFINED	**Private Memory Segment Descriptor Partitioning Select** 0b = (Default) The GLHMC_SDPART register is used for calculating the absolute SD Index to access within the Segment Table, and for checking whether or not *PMSDIDX* is within this function's allocated range. 1b = The GLHMC_PFPESDPART register is used for calculating the absolute SD Index to access within the Segment Table, and for checking whether or not *PMSDIDX* is within this function's allocated range.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	30:16	0x0	RSV	N/A	Reserved.
PMSDWR	31	0b	RW	UNDEFINED	**Private Memory Segment Descriptor Write/Read** 0b = Read operations. 1b = Write operations.

13.2.2.20.3 Private Memory Space Segment Descriptor Data Low - PFHMC_SDDATALOW_FPMAT (0x00100100; RW)

This register is used in conjunction with PFHMC_SDCMD and PFHMC_SDDATAHIGH to access the Host Memory Cache's segment table.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMSDVALID	0	0b	RW	UNDEFINED	**Private Memory Segment Descriptor Valid** Valid bit of an HMC segment descriptor table entry.
PMSDTYPE	1	0b	RW	UNDEFINED	**Private Memory Segment Descriptor Type** 0b = The Segment Descriptor is paged (the SD points to a the physical address of a host memory page that contains an array of Page Descriptors. 1b = The Segment Descriptor directly points the physical address of a physically contiguous 2 MB memory region.
PMSDBPCOUNT	11:2	0x0	RW	UNDEFINED	**Private Memory Segment Descriptor Backing Page Count** Backing Page count of an HMC segment descriptor table entry. Every SD entry in a given Function Private memory space must be set to 512, except the last SD. The last SD can have a value from 1 to 512. This field is used to calculate the end of the FPM space associated with a Segment Descriptor without having to read the valid bit for each individual PD entry.
PMSDDATALOW	31:12	0x0	RW	UNDEFINED	**Private Memory Segment Descriptor Data Low** Bits 31:12 bits of an HMC segment descriptor table entry.

13.2.2.20.4 Private Memory Space Segment Descriptor Data High - PFHMC_SDDATAHIGH_FPMAT (0x00100200; RW)

This register is used in conjunction with PFHMC_SDCMD and PFHMC_SDDATALOW to access the Host Memory Cache's segment table.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMSDDATAHIGH	31:0	0x0	RW	UNDEFINED	**Private Memory Segment Descriptor Data High** Most significant 32 bits of a segment descriptor.

13.2.2.20.5 Private Memory Space Page Descriptor Invalidate - PFHMC_PDINV_FPMAT (0x00100300; RW)

This register is used to invalidate cached HMC page descriptors that have been set to the invalid state by software.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMSDIDX	11:0	0x0	RW	UNDEFINED	**Private Memory Segment Descriptor Index** Relative Index of the HMC Segment Descriptor associated with the HMC Page Descriptor that is to be invalidated. For PFs, the actual index to be used to access the Segment Table is (*PMSDBASE* + *PMSDIDX*), where *PMSDBASE* is from the *PMSDBASE* field of the GLHMC_SDPART or GLHMC_PFPESDPART register associated with this function. If (*PMSDIDX* >= *PMSDSIZE*), the invalidate request is dropped, where *PMSDSIZE* is from the *PMSDSIZE* field of the GLHMC_SDPART or GLHMC_PFPESDPART register associated with this function. The *PMSDPARTSEL* bit of this register determines which SD Partition register is selected for the absolute SD index calculation and for the check that *PMSDIDX* is within the range allowed by this function. For Protocol Engine enabled VFs, the actual index to be used to invalidate a Segment Table entry is (*PMSDBASE* + *PMSDIDX*), where *PMSDBASE* is from the GLHMC_VFSDPART register associated with this function. For VFs, if (*PMSDIDX* >= GLHMC_VFSDPART.*PMSDSIZE*), the invalidate request is dropped. The *PMSDPARTSEL* bit is ignored in the GLHMC_VFPDINV registers. For the GLHMC_FWPDINV version of this register, this field is the absolute SD index, and the *PMSDPARTSEL* bit is ignored.
RESERVED	14:12	000b	RSV	N/A	Reserved.
PMSDPARTSEL	15	0b	RW	UNDEFINED	**Private Memory Segment Descriptor Partitioning Select** 0b = (Default) The GLHMC_SDPART register is used for calculating the absolute SD Index to invalidate within the Segment Table, and for checking whether or not *PMSDIDX* is within this function's allocated range. 1b = The GLHMC_PFPESDPART register is used for calculating the absolute SD Index to invalidate within the Segment Table, and for checking whether or not *PMSDIDX* is within this function's allocated range. This field is ignored in the GLHMC_VFPDINV and GLHMC_FWPDINV versions of this register.
PMPDIDX	24:16	0x0	RW	UNDEFINED	**Private Memory Page Descriptor Index** Index of the Page Descriptor within the Page Descriptor Page indicated by *PMSDIDX*.
RESERVED	31:25	0x0	RSV	N/A	Reserved.

13.2.2.20.6 Host Memory Cache Error Information Register - PFHMC_ERRORINFO_FPMAT (0x00100400; RW)

This register reports the errors detected by the Host Memory Cache. Errors reported through this register also can trigger interrupts through the *HMC_ERR* bit in the PFINT_ICR0 register.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMF_INDEX	4:0	0x0	RW	UNDEFINED	**Private Memory Function Index** This field reports the HMC Private Memory Function associated with the error. Writes to this field are ignored.
RESERVED	6:5	00b	RSV	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMF_ISVF	7	0b	RW	UNDEFINED	**Private Memory Function Is VF** 0b = The Private Memory Function reported in *PMF_INDEX* is associated with a PF. 1b = The Private Memory Function reported in *PMF_INDEX* is associated with a Protocol Engine enabled VF. Writes to this field are ignored.
HMC_ERROR_TYPE	11:8	0x0	RW	UNDEFINED	**HMC Error Type** This field reports the error type detected by the Host Memory Cache. The values are: 0 = Private Memory Function is not valid (the valid bit is clear in the GLHMC_VFPMFMAP register associated with the PMF). 1 = Invalid Private Memory Function index for a Protocol Engine enabled VF in GLHMC_VFPMFMAP. (i.e., The PMF Index programmed in the GLHMC_VFPMFMAP register is < 8 or > 40. This is a firmware error.) 2 = Invalid PF for a Protocol Engine enabled VF in GLHMC_VFPMFTABLE. (i.e., the parent PF index programmed in the GLHMC_VFPMFMAP register did not match the PF index received in the HMC transaction. This is a firmware error.) 3 = Reserved. In predecessor devices it was used to be an indication for Invalid LAN Queue Index or FCoE VF Index. (i.e., The absolute LAN Queue Index or FCoE VF Index received in the HMC transaction was less than the LAN Queue Index Base register or FCoE VF Index Base register associated with the PF Index received in the HMC transaction.) 4 = Index to big error: Indication that Object Index from transaction was larger than the value specified in the object's GLHMC_*CNT register. 5 = Private Memory Address Extends beyond the limits of the Segment Descriptors assigned to the PCIe function. 6 = Segment Descriptor Invalid. 7 = Segment Descriptor Too Small (only applies to Direct Mapped SDs). 8 = Page Descriptor Invalid. 9 = Received Unsupported Request (UR) Completion from PCIe read of object. 10 = Reserved. In predecessor devices this was used to be an indication that the valid bit in PFLAN_QALLOC_PMAT[PF] or PF_VT_PFALLOC_PMAT[PF] register was not set. 11 = An invalid object type was detected. 12 = Reserved. In predecessor devices it was used to be an indication for Object Index for an FCoE DDP Context object was larger than the size specified in the *PFFCDSIZE* or *VFFCDSIZE* field of the corresponding PFQF_CTL_0_PMAT register, or the object index for an FCoE Filter object was larger than the sum of the sizes specified in the *PFFCHSIZE* and *PFFCDSIZE* fields or *VFFCHSIZE* and *VFFCDSIZE* fields of the corresponding PFQF_CTL_0_PMAT register. All other values are reserved. The PFHMC_ERRORDATA register can be read to determine the LAN Queue index or FCoE VF index associated with error type 3. The PFHMC_ERRORDATA register can be read to determine the HMC object index associated with error types 4 and 12. The PFHMC_ERRORDATA register can be read to determine the HMC function relative *SD_Index* and *PD_Index* associated with error types 5 through 9.
RESERVED	15:12	0x0	RSV	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
HMC_OBJECT_TYPE	20:16	0x0	RW	UNDEFINED	**HMC Object Type** Specifies the object type associated with the error. The encodings for the object type are as follows: 0x00 = QP_CNTXT 0x01 = ARP_TBL_ENTRY 0x02 = TXFIFO 0x03 = IRRQ 0x04 = MRTE 0x05 = PBLE 0x06 = CQ_CNTXT 0x07 = SRQ_CNTXT 0x08 = APBVT_INUSE 0x09 = FSI_ADR_VCTR 0x0A = FSI_MCAST_GRP 0x0B = XF_FL 0x0C = Q1_FL 0x0D = TIMER 0x10 = LAN_TXQ_CNTXT 0x11 = LAN_RXQ_CNTXT 0x12 = FCOE_CNTXT 0x13 = FCOE_DDP_HTE 0x16 = QUAD_HTE 0x19 = PD All other values are reserved.
RESERVED	30:21	0x0	RSV	N/A	Reserved.
ERROR_DETECTED	31	0b	RW	UNDEFINED	**Error Detected** This field is set to 1b when a new error has been detected by the HMC. No subsequent errors are recorded until this field is written with a value of 0b. Write of a 0b to this field clears the error and allow a subsequent error to be reported. Writes of 1b to this field are ignored.

13.2.2.20.7 Host Memory Cache Error Data Register - PFHMC_ERRORDATA_FPMAT (0x00100500; RO)

This register reports the HMC function relative *SD_Index* and *PD_Index* or HMC object index related to an error detected by the Host Memory Cache.

Field	Bit(s)	Init.	Type	CFG Policy	Description
HMC_ERROR_DATA	29:0	0x0	RO	N/A	**HMC Error Data** This field reports either the HMC function relative *SD_Index*, *PD_Index*, LAN Queue index, FCoE VF index, or HMC object index associated with the error reported in the PFHMC_ERRORINFO register. FCoE reporting is not valid anymore in this device. When PFHMC_ERRORINFO.*HMC_ERROR_TYPE* is 3: • *HMC_ERROR_DATA*[27:0] reports the LAN Queue index or FCoE VF index associated with the error. • *HMC_ERROR_DATA*[29:28] should be zero. • This option is not valid anymore in this device. When PFHMC_ERRORINFO.*HMC_ERROR_TYPE* is 4 or 12: • *HMC_ERROR_DATA*[27:0] reports the object index associated with the error. • *HMC_ERROR_DATA*[29:28] should be zero. When PFHMC_ERRORINFO.*HMC_ERROR_TYPE* is 5 through 9: • *HMC_ERROR_DATA*[29:9] reports the HMC function relative *SD_Index* and *PD_Index* associated with the error detected by the HMC. • *HMC_ERROR_DATA*[29:18] is set to HMC function relative *SD_Index* for the affected HMC function. • *HMC_ERROR_DATA*[17:9] is set to the *PD_Index*. • HMC_ERROR_DATA[8:0] are reserved for error types 5 through 9. When PFHMC_ERRORINFO.*HMC_ERROR_TYPE* is 0 through 2, or 10 through 1: • *HMC_ERROR_DATA*[29:0] is not valid, and should be zero.
RESERVED	31:30	00b	RSV	N/A	Reserved.

13.2.2.20.8 Private Memory Segment Table Partitioning Registers - GLHMC_SDPART_FPMAT[n] (0x00100800 + 0x4*n, n=0...7; RO)

This register is used to partition the shared Host Memory Cache segment table.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMSDBASE	11:0	0x0	RW	UNDEFINED	**Private Memory Segment Descriptor Base** Base segment table index for the function n.
RESERVED	15:12	0x0	RSV	N/A	Reserved.
PMSDSIZE	28:16	0x0	RW	UNDEFINED	**Private Memory Segment Descriptor Size** Number of valid segment table entries for the function n.
RESERVED	31:29	000b	RSV	N/A	Reserved.

Note: This register must be read only in the PF CSR Space.

13.2.2.20.9 Private Memory Segment Table Partitioning Registers - GLHMC_PFPESDPART_FPMAT[n] (0x00100880 + 0x4*n, n=0...7; RO)

This register is used to partition the shared Host Memory Cache segment table.

Field definitions are the same as those defined in Section 13.2.2.20.41.

Note: This register must be read only in the PF CSR Space.

13.2.2.20.10 Private Memory PE Hash Table Entry Object Size - GLHMC_PEHTEOBJSZ_FPMAT (0x0010202C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEHTEOBJSZ	3:0	0x6	RO	N/A	**Protocol Engine Hash Table Entry Object Size** Used to calculate the amount of memory that is required to be allocated for Host Memory Cache Protocol Engine Hash Table Entry objects. 0x6 = 64 bytes
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.20.11 Private Memory Protocol Engine Hash Entry Max - GLHMC_PEHTMAX_FPMAT (0x00102030; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEHTMAX	20:0	0x140000	RO	N/A	**Private Memory PE Hash Table Entry Max** Reports that maximum number of hash table entries supported by the Host Memory Cache.
RESERVED	31:21	0x0	RSV	N/A	Reserved.

Note: This register is different from GLHMC_PEHTMAX because the PE changed to account for multicast groups.

13.2.2.20.12 Private Memory Space Segment Descriptor Data Low - GLHMC_FWSDDATALOW_FPMAT (0x00102074; RO)

This register is used in conjunction with PFHMC_SDCMD and PFHMC_SDDATAHIGH to access the Host Memory Cache's segment table.

Field definitions are the same as those defined in Section 13.2.2.20.36.

13.2.2.20.13 Private Memory Space Segment Descriptor Data High - GLHMC_FWSDDATAHIGH_FPMAT (0x00102078; RO)

This register is used in conjunction with PFHMC_SDCMD and PFHMC_SDDATALOW to access the Host Memory Cache's segment table.

Field definitions are the same as those defined in Section 13.2.2.20.37.

13.2.2.20.14 Private Memory Space Page Descriptor Invalidate - GLHMC_FWPDINV_FPMAT (0x0010207C; RO)

This register is used to invalidate cached HMC page descriptors that have been set to the invalid state by software.

Field definitions are the same as those defined in Section 13.2.2.20.38.

13.2.2.20.15 FPM PE Hash Table Entry Base - GLHMC_PEHTEBASE_FPMAT[n] (0x00104600 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPEHTEBASE	23:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Engine Hash Table Entry Base** Reports the Function Private Memory space base address for the Protocol Engine Hash Table Entry objects in 512-byte increments. In other words, the value in this registers must be multiplied by 512 to get the actual address out of the 8 GB FPM address space. This register is updated by hardware when the Commit FPM Values CQP operation is performed. Other than for debug purposes, this register should be treated as Read Only.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

Note: 16 instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.16 FPM PE Hash Table Object Count - GLHMC_PEHTCNT_FPMAT[n] (0x00104700 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPEHTCNT	28:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Engine Hash Table Object Count** Used to set the Function Private Memory space size for the Protocol Engine Hash Table objects. The associated base register is updated after the Commit FPM Values CQP operation is performed to indicate to hardware that all of the FPM size registers have been set properly and the FPM map should be recomputed.
RESERVED	31:29	000b	RSV	N/A	Reserved.

Note: 16 instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.17 Private Memory Space VF Segment Descriptor Data Low - GLHMC_VFSDDATALOW_FPMAT[n] (0x00108100 + 0x4*n, n=0...31; RO)

This register is used in conjunction with PFHMC_SDCMD and PFHMC_SDDATAHIGH to access the Host Memory Cache's segment table.

Field definitions are the same as those defined in Section 13.2.2.20.36.

Note: 16 instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.18 Private Memory Space VF Segment Descriptor Data High - GLHMC_VFSDDATAHIGH_FPMAT[n] (0x00108200 + 0x4*n, n=0...31; RO)

This register is used in conjunction with PFHMC_SDCMD and PFHMC_SDDATALOW to access the Host Memory Cache's segment table.

Field definitions are the same as those defined in Section 13.2.2.20.37.

Note: 16 instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.19 Private Memory Space Page Descriptor Invalidate - GLHMC_VFPDINV_FPMAT[n] (0x00108300 + 0x4*n, n=0...31; RO)

This register is used to invalidate cached HMC page descriptors that have been set to the invalid state by software.

Field definitions are the same as those defined in Section 13.2.2.20.38.

Note: 16 instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.20 Private Memory Segment Table Partitioning Registers - GLHMC_VFSDPART_FPMAT[n] (0x00108800 + 0x4*n, n=0...31; RO)

This register is used to partition the shared Host Memory Cache segment table.

Field definitions are the same as those defined in Section 13.2.2.20.41.

Note: This register must be read only in the PF CSR Space.

13.2.2.20.21 FPM PE Hash Table Entry Base - GLHMC_VFPEHTEBASE_FPMAT[n] (0x0010C600 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.83.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.22 FPM PE Hash Table Object Count - GLHMC_VFPEHTCNT_FPMAT[n] (0x0010C700 + 0x4*n, n=0...31; RW)

Field definitions are the same as those defined in Section 13.2.2.20.84.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.23 PDOC Cache Attributes - GLPDOC_CACHESIZE_FPMAT (0x00110088; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
WORD_SIZE	7:0	0x40	RO	N/A	**Word Size** The cache line size in bytes.
SETS	19:8	0x40	RO	N/A	**Sets** The number of cache sets.
WAYS	23:20	0x8	RO	N/A	**Ways** The number of cache ways.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.20.24 Private Memory CQ Doorbell Partition Registers - GLHMC_VFDBCQPART[n] (0x00502E00 + 0x4*n, n=0...31; RO)

These registers are used to partition the shared doorbell array for Protocol Engine Completion Queues PCIe PFs.

Field definitions are the same as those defined in Section 13.2.2.20.26.

Note: This register must be read only in the PF CSR Space.

13.2.2.20.25 Private Memory CEQ Partitioning Registers - GLHMC_VFCEQPART[n] (0x00502F00 + 0x4*n, n=0...31; RO)

This register is used to partition the shared pool of Protocol Engine Completion Event Queues for PCIe PFs.

Field definitions are the same as those defined in Section 13.2.2.20.27.

Note: This register must be read only in the PF CSR Space.

13.2.2.20.26 Private Memory CQ Doorbell Partition Registers - GLHMC_DBCQPART[n] (0x00503180 + 0x4*n, n=0...7; RO)

These registers are used to partition the shared doorbell array for Protocol Engine Completion Queues PCIe PFs.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMDBCQBASE	13:0	0x0	RW	UNDEFINED	**Private Memory Doorbell CQ Base** Base CQ doorbell array index for the Host Memory Cache PCI function n in multiples of 64 CQs.
RESERVED	15:14	00b	RSV	N/A	Reserved.
PMDBCQSIZE	30:16	0x0	RW	UNDEFINED	**Private Memory Doorbell CQ Size** Number of valid doorbell array elements for the Host Memory Cache PCI function n in increments of 64 CQs.
RESERVED	31	0b	RSV	N/A	Reserved.

Note: This register must be read only in the PF CSR Space.

13.2.2.20.27 Private Memory CEQ Partitioning Registers - GLHMC_CEQPART[n] (0x005031C0 + 0x4*n, n=0...7; RO)

This register is used to partition the shared pool of Protocol Engine Completion Event Queues for PCIe PFs.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMCEQBASE	9:0	0x0	RW	UNDEFINED	**Private Memory CEQ Base** Base CEQ index for the function n.
RESERVED	15:10	0x0	RSV	N/A	Reserved.
PMCEQSIZE	25:16	0x0	RW	UNDEFINED	**Private Memory CEQ Size** Number of valid CEQs index for the function n.
RESERVED	31:26	0x0	RSV	N/A	Reserved.

Note: This register must be read only in the PF CSR Space.

13.2.2.20.28 Private Memory QP Doorbell Partition Registers - GLHMC_DBQPPART[n] (0x005044C0 + 0x4*n, n=0...7; RO)

These registers are used to partition the shared doorbell array for Protocol Engine Queue Pairs associated with PCIe PFs.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMDBQPBASE	13:0	0x0	RW	UNDEFINED	**Private Memory Doorbell QP Base** Base QP doorbell array index for the Host Memory Cache PCI function n in multiples of 128 QPs.
RESERVED	15:14	00b	RSV	N/A	Reserved.
PMDBQPSIZE	30:16	0x0	RW	UNDEFINED	**Private Memory Doorbell QP Size** Number of valid doorbell array elements for the Host Memory Cache PCI function n in increments of 128 QPs.
RESERVED	31	0b	RSV	N/A	Reserved.

Note: This register must be read only in the PF CSR Space. This register is configured by CQP firmware instead of from NVRAM due to the Protocol Engine clock gating at chip initialization.

13.2.2.20.29 Private Memory VF QP Doorbell Partition Registers - GLHMC_VFDBQPPART[n] (0x00504520 + 0x4*n, n=0...31; RO)

These registers are used to partition the shared doorbell array for Protocol Engine Queue Pairs associated with PCIe PFs.

Field definitions are the same as those defined in Section 13.2.2.20.28.

Note: This register must be read only in the PF CSR Space. This register is configured by CQP firmware instead of from NVRAM due to the Protocol Engine clock gating at chip initialization.

13.2.2.20.30 PEOC0 Cache Attributes - GLPEOC0_CACHESIZE (0x005140A8; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
WORD_SIZE	7:0	0x40	RO	N/A	**Word Size** The cache line size in bytes.
SETS	19:8	0x200	RO	N/A	**Sets** The number of cache sets.
WAYS	23:20	0x8	RO	N/A	**Ways** The number of cache ways.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.20.31 PEOC1 Cache Attributes - GLPEOC1_CACHESIZE (0x005160A8; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
WORD_SIZE	7:0	0x40	RO	N/A	**Word Size** The cache line size in bytes.
SETS	19:8	0x200	RO	N/A	**Sets** The number of cache sets.
WAYS	23:20	0x8	RO	N/A	**Ways** The number of cache ways.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.20.32 PBLOC0 Cache Attributes - GLPBLOC0_CACHESIZE (0x00518074; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
WORD_SIZE	7:0	0x40	RO	N/A	**Word Size** The cache line size in bytes.
SETS	19:8	0x100	RO	N/A	**Sets** The number of cache sets.
WAYS	23:20	0x8	RO	N/A	**Ways** The number of cache ways.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.20.33 PBLOC1 Cache Attributes - GLPBLOC1_CACHESIZE (0x0051A074; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
WORD_SIZE	7:0	0x40	RO	N/A	**Word Size** The cache line size in bytes.
SETS	19:8	0x100	RO	N/A	**Sets** The number of cache sets.
WAYS	23:20	0x8	RO	N/A	**Ways** The number of cache ways.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.20.34 MDOC Cache Attributes - GLMDOC_CACHESIZE (0x0051C06C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
WORD_SIZE	7:0	0x80	RO	N/A	**Word Size** The cache line size in bytes.
SETS	19:8	0x200	RO	N/A	**Sets** The number of cache sets.
WAYS	23:20	0x8	RO	N/A	**Ways** The number of cache ways.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.20.35 Private Memory Space Segment Descriptor Command - PFHMC_SDCMD (0x00520000; RW)

This register is used to access the Host Memory Cache's segment table. The HMC's segment table is partitioned per Private Memory Function (PMF), and this register is only allowed to access the Segment Table entries allocated to the PMF associated with this register via the SD Partition register, GLHMC_SDPART.

For read operations, PFHMC_SDCMD must be written with *PMSDWR* set to 0b and *PMSDIDX* set to the segment table index to be read. After the write to PFHMC_SDCMD completes, PFHMC_SDDATALOW and PFHMC_SDDATAHIGH can be read to retrieve the segment descriptor contents.

For write operations, PFHMC_SDDATALOW and PFHMC_SDDATAHIGH must be written before writing PFHMC_SDCMD with *PMSDWR* set to 1b and *PMSDIDX* set to the segment table index to be written.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMSDIDX	11:0	0x0	RW	UNDEFINED	**Private Memory Segment Descriptor Index** Relative Index of the HMC Segment Descriptor to be read or written. The actual absolute index to be used to access the Segment Table is (*PMSDBASE* + *PMSDIDX*), where *PMSDBASE* is from the *PMSDBASE* field of the GLHMC_SDPART or GLHMC_PFPESDPART register associated with this function. On write operations, if (*PMSDIDX* >= *PMSDSIZE*), the write is dropped, where *PMSDSIZE* is from the *PMSDSIZE* field of the GLHMC_SDPART or GLHMC_PFPESDPART register associated with this function. The *PMSDPARTSEL* bit of this register determines which SD Partition register is selected for the absolute *SD_Index* calculation and for the check that *PMSDIDX* is within the range allowed by this function.
RESERVED	14:12	000b	RSV	N/A	Reserved.
PMSDPARTSEL	15	0b	RW	UNDEFINED	**Private Memory Segment Descriptor Partitioning Select** 0b = (Default) The GLHMC_SDPART register is used for calculating the absolute SD Index to access within the Segment Table, and for checking whether or not *PMSDIDX* is within this function's allocated range. 1b = The GLHMC_PFPESDPART register is used for calculating the absolute SD Index to access within the Segment Table, and for checking whether or not *PMSDIDX* is within this function's allocated range.
RESERVED	30:16	0x0	RSV	N/A	Reserved.
PMSDWR	31	0b	RW	UNDEFINED	**Private Memory Segment Descriptor Write/Read** 0b = Read operations. 1b = Write operations.

13.2.2.20.36 Private Memory Space Segment Descriptor Data Low - PFHMC_SDDATALOW (0x00520100; RW)

This register is used in conjunction with PFHMC_SDCMD and PFHMC_SDDATAHIGH to access the Host Memory Cache's segment table.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMSDVALID	0	0b	RW	UNDEFINED	**Private Memory Segment Descriptor Valid** Valid bit of an HMC segment descriptor table entry.
PMSDTYPE	1	0b	RW	UNDEFINED	**Private Memory Segment Descriptor Type** 0b = The Segment Descriptor is paged (the SD points to a the physical address of a host memory page that contains an array of Page Descriptors. 1b = The Segment Descriptor directly points the physical address of a physically contiguous 2 MB memory region.
PMSDBPCOUNT	11:2	0x0	RW	UNDEFINED	**Private Memory Segment Descriptor Backing Page Count** Backing Page count of an HMC segment descriptor table entry. Every SD entry in a given Function Private memory space must be set to 512, except the last SD. The last SD can have a value from 1 to 512. This field is used to calculate the end of the FPM space associated with a Segment Descriptor without having to read the valid bit for each individual PD entry
PMSDDATALOW	31:12	0x0	RW	UNDEFINED	**Private Memory Segment Descriptor Data Low** Bits 31:12 bits of an HMC segment descriptor table entry.

13.2.2.20.37 Private Memory Space Segment Descriptor Data High - PFHMC_SDDATAHIGH (0x00520200; RW)

This register is used in conjunction with PFHMC_SDCMD and PFHMC_SDDATALOW to access the Host Memory Cache's segment table.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMSDDATAHIGH	31:0	0x0	RW	UNDEFINED	**Private Memory Segment Descriptor Data High** Most significant 32 bits of a segment descriptor.

13.2.2.20.38 Private Memory Space Page Descriptor Invalidate - PFHMC_PDINV (0x00520300; RW)

This register is used to invalidate cached HMC page descriptors that have been set to the invalid state by software.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMSDIDX	11:0	0x0	RW	UNDEFINED	**Private Memory Segment Descriptor Index** Relative Index of the HMC Segment Descriptor associated with the HMC Page Descriptor that is to be invalidated. For PFs, the actual index to be used to access the Segment Table is (*PMSDBASE* + *PMSDIDX*), where *PMSDBASE* is from the *PMSDBASE* field of the GLHMC_SDPART or GLHMC_PFPESDPART register associated with this function. If (*PMSDIDX* >= *PMSDSIZE*), the invalidate request is dropped, where *PMSDSIZE* is from the *PMSDSIZE* field of the GLHMC_SDPART or GLHMC_PFPESDPART register associated with this function. The *PMSDPARTSEL* bit of this register determines which SD Partition register is selected for the absolute SD index calculation and for the check that *PMSDIDX* is within the range allowed by this function. For Protocol Engine enabled VFs, the actual index to be used to invalidate a Segment Table entry is (*PMSDBASE* + *PMSDIDX*), where *PMSDBASE* is from the GLHMC_VFSDPART register associated with this function. For VFs, if (*PMSDIDX* >= GLHMC_VFSDPART.*PMSDSIZE*), the invalidate request will be dropped. The *PMSDPARTSEL* bit is ignored in the GLHMC_VFPDINV registers. For the GLHMC_FWPDINV version of this register, this field is the absolute SD index, and the *PMSDPARTSEL* bit is ignored.
RESERVED	14:12	000b	RSV	N/A	Reserved.
PMSDPARTSEL	15	0b	RW	UNDEFINED	**Private Memory Segment Descriptor Partitioning Select** 0b = (Default) The GLHMC_SDPART register is used for calculating the absolute SD Index to invalidate within the Segment Table, and for checking whether or not *PMSDIDX* is within this function's allocated range. 1b = The GLHMC_PFPESDPART register is used for calculating the absolute SD Index to invalidate within the Segment Table, and for checking whether or not *PMSDIDX* is within this function's allocated range. This field is ignored in the GLHMC_VFPDINV and GLHMC_FWPDINV versions of this register.
PMPDIDX	24:16	0x0	RW	UNDEFINED	**Private Memory Page Descriptor Index** Index of the Page Descriptor within the Page Descriptor Page indicated by *PMSDIDX*.
RESERVED	31:25	0x0	RSV	N/A	Reserved.

13.2.2.20.39 Host Memory Cache Error Information Register - PFHMC_ERRORINFO (0x00520400; RW)

This register reports the errors detected by the Host Memory Cache. Errors reported through this register also can trigger interrupts through the *HMC_ERR* bit in the PFINT_ICR0 register.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMF_INDEX	4:0	0x0	RW	UNDEFINED	**Private Memory Function Index** This field reports the HMC Private Memory Function associated with the error. Writes to this field are ignored.
RESERVED	6:5	00b	RSV	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMF_ISVF	7	0b	RW	UNDEFINED	**Private Memory Function Is VF** 0b = The Private Memory Function reported in *PMF_INDEX* is associated with a PF. 1b = The Private Memory Function reported in *PMF_INDEX* is associated with a Protocol Engine enabled VF. Writes to this field are ignored.
HMC_ERROR_TYPE	11:8	0x0	RW	UNDEFINED	**HMC Error Type** This field reports the error type detected by the Host Memory Cache. The values are: 0 = Private Memory Function is not valid (the valid bit is clear in the GLHMC_VFPMFMAP register associated with the PMF) 1 = Invalid Private Memory Function index for a Protocol Engine enabled VF in GLHMC_VFPMFMAP. (i.e., The PMF Index programmed in the GLHMC_VFPMFMAP register is < 8 or > 39. This is a firmware error.) 2 = Invalid PF for a Protocol Engine enabled VF in GLHMC_VFPMFTABLE. (i.e., the parent PF index programmed in the GLHMC_VFPMFMAP register did not match the PF index received in the HMC transaction. This is a firmware error. 3= Reserved. In predecessor devices it was used to be an indication for Invalid LAN Queue Index or FCoE VF Index. (i.e., The absolute LAN Queue Index or FCoE VF Index received in the HMC transaction was less than the LAN Queue Index Base register or FCoE VF Index Base register associated with the PF Index received in the HMC transaction.) 4 = Index to big error: Indication that Object Index from transaction was larger than the value specified in the object's GLHMC_*CNT register. 5 = Private Memory Address Extends beyond the limits of the Segment Descriptors assigned to the PCIe function. 6 = Segment Descriptor Invalid. 7 = Segment Descriptor Too Small (only applies to Direct Mapped SDs). 8 = Page Descriptor Invalid. 9 = Received Unsupported Request (UR) Completion from PCIe read of object. 10 = Reserved. In predecessor devices this was used to be an indication that the valid bit in PFLAN_QALLOC_PMAT[PF] or PF_VT_PFALLOC_PMAT[PF] register was not set. 11 = An invalid object type was detected. 12 = Reserved. In predecessor devices it was used to be an indication for Object Index for an FCoE DDP Context object was larger than the size specified in the *PFFCDSIZE* or *VFFCDSIZE* field of the corresponding PFQF_CTL_0_PMAT register, or the object index for an FCoE Filter object was larger than the sum of the sizes specified in the *PFFCHSIZE* and *PFFCDSIZE* fields or *VFFCHSIZE* and *VFFCDSIZE* fields of the corresponding PFQF_CTL_0_PMAT register. All other values are reserved. The PFHMC_ERRORDATA register can be read to determine the LAN Queue index or FCoE VF index associated with error type 3. The PFHMC_ERRORDATA register can be read to determine the HMC object index associated with error types 4 and 12. The PFHMC_ERRORDATA register can be read to determine the HMC function relative *SD_Index* and *PD_Index* associated with error types 5 through 9.
RESERVED	15:12	0x0	RSV	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
HMC_OBJECT_TYPE	20:16	0x0	RW	UNDEFINED	**HMC Object Type** Specifies the object type associated with the error. The encodings for the object type are as follows: 0x00 = QP_CNTXT 0x01 = ARP_TBL_ENTRY 0x02 = TXFIFO 0x03 = IRRQ 0x04 = MRTE 0x05 = PBLE 0x06 = CQ_CNTXT 0x07 = SRQ_CNTXT 0x08 = APBVT_INUSE 0x09 = FSI_ADR_VCTR 0x0A = FSI_MCAST_GRP 0x0B = XF_FL 0x0C = Q1_FL 0x0D = TIMER 0x10 = LAN_TXQ_CNTXT 0x11 = LAN_RXQ_CNTXT 0x12 = FCOE_CNTXT 0x13 = FCOE_DDP_HTE 0x16 = QUAD_HTE 0x19 = PD All other values are reserved.
RESERVED	30:21	0x0	RSV	N/A	Reserved.
ERROR_DETECTED	31	0b	RW	UNDEFINED	**Error Detected** This field is set to 1b when a new error has been detected by the HMC. No subsequent errors will be recorded until this field is written with a value of 0b. Writes of a 0 to this register clears the error and allow a subsequent error to be reported. Writes of 1 to this field are ignored.

13.2.2.20.40 Host Memory Cache Error Data Register - PFHMC_ERRORDATA (0x00520500; RO)

This register reports the HMC function relative SD_Index and PD_Index or HMC object index related to an error detected by the Host Memory Cache.

Field	Bit(s)	Init.	Type	CFG Policy	Description
HMC_ERROR_DATA	29:0	0x0	RO	N/A	**HMC Error Data** This field reports either the HMC function relative *SD_Index*, *PD_Index*, LAN Queue index, FCoE VF index, or HMC object index associated with the error reported in the PFHMC_ERRORINFO register. FCoE reporting is not valid anymore in this device. When PFHMC_ERRORINFO.*HMC_ERROR_TYPE* is 3: • *HMC_ERROR_DATA*[27:0] reports the LAN Queue index or FCoE VF index associated with the error. • *HMC_ERROR_DATA*[29:28] should be zero. • This option is not valid anymore in this device. When PFHMC_ERRORINFO.*HMC_ERROR_TYPE* is 4 or 12: • *HMC_ERROR_DATA*[27:0] reports the object index associated with the error. • *HMC_ERROR_DATA*[29:28] should be zero. When PFHMC_ERRORINFO.*HMC_ERROR_TYPE* is 5 through 9: • *HMC_ERROR_DATA*[29:9] reports the HMC function relative *SD_Index* and *PD_Index* associated with the error detected by the HMC. • *HMC_ERROR_DATA*[29:18] is set to HMC function relative *SD_Index* for the affected HMC function. • *HMC_ERROR_DATA*[17:9] is set to the *PD_Index*. • HMC_ERROR_DATA[8:0] are reserved for error types 5 through 9. When PFHMC_ERRORINFO.*HMC_ERROR_TYPE* is 0 through 2, or 10 through 1: • *HMC_ERROR_DATA*[29:0] is not valid, and should be zero.
RESERVED	31:30	0x0	RSV	N/A	Reserved.

13.2.2.20.41 Private Memory Segment Table Partitioning Registers - GLHMC_SDPART[n] (0x00520800 + 0x4*n, n=0...7; RO)

This register is used to partition the shared Host Memory Cache segment table.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMSDBASE	11:0	0x0	RW	UNDEFINED	**Private Memory Segment Descriptor Base** Base segment table index for the function n.
RESERVED	15:12	0x0	RSV	N/A	Reserved.
PMSDSIZE	28:16	0x0	RW	UNDEFINED	**Private Memory Segment Descriptor Size** Number of valid segment table entries for the function n.
RESERVED	31:29	0x0	RSV	N/A	Reserved.

Note: This register must be read only in the PF CSR Space.

13.2.2.20.42 Private Memory Segment Table Partitioning Registers - GLHMC_PFPESDPART[n] (0x00520880 + 0x4*n, n=0...7; RO)

This register is used to partition the shared Host Memory Cache segment table.

Field definitions are the same as those defined in Section 13.2.2.20.41.

Note: This register must be read only in the PF CSR Space.

13.2.2.20.43 Private Memory Protocol Engine Header Max - GLHMC_PEHDROBJSZ (0x00522004; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEHDROBJSZ	3:0	0x6	RO	N/A	**Private Memory Protocol Engine Header Object Size** Used to calculate the amount of memory that is required to be allocated for Host Memory Cache Protocol Engine Header objects. 0x6 = 64 bytes
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.20.44 FPM PE Header Object Count - GLHMC_PEHDRMAX (0x00522008; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEHDRMAX	18:0	0x40000	RO	N/A	**Private Memory Protocol Engine Header Max** Reports the maximum number of Protocol Engine Metadata objects supported by the Host Memory Cache.
RESERVED	31:19	0x0	RSV	N/A	Reserved.

13.2.2.20.45 Private Memory PE Metadata Object Size - GLHMC_PEMDOBJSZ (0x0052200C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEMDOBJSZ	3:0	0x4	RO	N/A	**Private Memory Protocol Engine Metadata Object Size** Used to calculate the amount of memory that is required to be allocated for Host Memory Cache Protocol Engine Metadata objects. 0x4 = 16 bytes
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.20.46 Private Memory Protocol Engine Metadata Max - GLHMC_PEMDMAX (0x00522010; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEMDMAX	23:0	0x800000	RO	N/A	**Private Memory Protocol Engine Metadata Max** Reports the maximum number of Protocol Engine Metadata objects supported by the Host Memory Cache.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.20.47 Private Memory PE Out of Order Send Completion Object Size - GLHMC_PEOOISCOBJSZ (0x00522014; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEOOISCOBJSZ	3:0	0x5	RO	N/A	**Private Memory Protocol Engine Out-of-Order Send Completion Object Size** Used to calculate the amount of memory that is required to be allocated for Host Memory Cache Protocol Engine Out-of-Order Send Completion objects. 0x5 = 32 bytes
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.20.48 Private Memory Protocol Engine Out of Order Send Completion Max - GLHMC_PEOOISCMAX (0x00522018; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEOOISCMAX	18:0	0x40000	RO	N/A	**Private Memory Protocol Engine Out-of-Order Send Completion Max** Reports the maximum number of Protocol Engine Out-of-Order Send Completions supported by the Host Memory Cache.
RESERVED	31:19	0x0	RSV	N/A	Reserved.

13.2.2.20.49 Private Memory PE QP Object Size - GLHMC_PEQPOBJSZ (0x0052201C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEQPOBJSZ	3:0	0x9	RO	N/A	**Private Memory Protocol Engine QP Object Size** Used to calculate the amount of memory that is required to be allocated for Host Memory Cache Protocol Engine Queue Pair objects. 0x9 = 512 bytes
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.20.50 Private Memory PE CQ Object Size - GLHMC_PECQOBJSZ (0x00522020; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPECQOBJSZ	3:0	0x6	RO	N/A	**Private Memory Protocol Engine CQ Object Size** Used to calculate the amount of memory that is required to be allocated for Host Memory Cache Protocol Engine Completion Queue objects. 0x6 = 64 bytes
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.20.51 Private Memory PE Hash Table Entry Object Size - GLHMC_PEHTEOBJSZ (0x0052202C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEHTEOBJSZ	3:0	0x6	RO	N/A	**Private Memory Protocol Engine Hash Table Entry Object Size** Used to calculate the amount of memory that is required to be allocated for Host Memory Cache Protocol Engine Hash Table Entry objects. 0x6 = 64 bytes
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.20.52 Private Memory Protocol Engine Hash Entry Max - GLHMC_PEHTMAX (0x00522030; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEHTMAX	20:0	0x14A000	RO	N/A	**Private Memory Protocol Engine Hash Table Max** Reports that maximum number of Hash Table entries supported by the Host Memory Cache.
RESERVED	31:21	0x0	RSV	N/A	Reserved.

13.2.2.20.53 Private Memory PE ARP Table Entry Object Size - GLHMC_PEARPOBJSZ (0x00522034; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEARPOBJSZ	2:0	0x4	RO	N/A	**Private Memory Protocol Engine ARP Object Size** Used to calculate the amount of memory that is required to be allocated for Host Memory Cache ARP Table Entry objects. 0x4 = 16 bytes
RESERVED	31:3	0x0	RSV	N/A	Reserved.

13.2.2.20.54 Private Memory Protocol Engine ARP Table Entry Max - GLHMC_PEARPMAX (0x00522038; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEARPMAX	16:0	0x10000	RO	N/A	**Private Memory Protocol Engine ARP Max** Reports that maximum number of ARP Table entries supported by the Host Memory Cache.
RESERVED	31:17	0x0	RSV	N/A	Reserved.

13.2.2.20.55 Private Memory PE Memory Region Table Entry Object Size - GLHMC_PEMROBJSZ (0x0052203C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEMROBJSZ	3:0	0x5	RO	N/A	**Private Memory Protocol Engine Memory Region Object Size** Used to calculate the amount of memory that is required to be allocated for Host Memory Cache Memory Region Table Entry objects. 0x5 = 32 bytes
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.20.56 Private Memory Protocol Engine Memory Registration Max - GLHMC_PEMRMAX (0x00522040; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEMRMAX	22:0	0x400000	RO	N/A	**Private Memory Protocol Engine Memory Region Max** Reports that maximum number of Memory Registration Table entries supported by the Host Memory Cache.
RESERVED	31:23	0x0	RSV	N/A	Reserved.

13.2.2.20.57 Private Memory PE Xmit FIFO Object Size - GLHMC_PEXFOBJSZ (0x00522044; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEXFOBJSZ	3:0	0x5	RO	N/A	**Private Memory Protocol Engine Transmit FIFO Object Size** Used to calculate the amount of memory that is required to be allocated for Host Memory Cache Xmit FIFO objects. 0x5 = 32 bytes
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.20.58 Private Memory Protocol Engine Transmit FIFO Entry Max - GLHMC_PEXFMAX (0x00522048; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEXFMAX	27:0	0x8000000	RO	N/A	**Private Memory Protocol Engine Transmit FIFO Max** Reports that maximum number of Transmit FIFO entries supported by the Host Memory Cache.
RESERVED	31:28	0x0	RSV	N/A	Reserved.

13.2.2.20.59 Private Memory Protocol Engine Transmit FIFO Free List Max - GLHMC_PEXFFLMAX (0x0052204C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEXFFLMAX	27:0	0x2000000	RO	N/A	**Private Memory Protocol Engine Transmit FIFO Free List Max** Reports that maximum number of Transmit FIFO Free List entries supported by the Host Memory Cache.
RESERVED	31:28	0x0	RSV	N/A	Reserved.

13.2.2.20.60 Private Memory PE IRRQ Object Size - GLHMC_PEQ1OBJSZ (0x00522050; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEQ1OBJSZ	3:0	0x6	RO	N/A	**Private Memory Protocol Engine Q1 Object Size** Used to calculate the amount of memory that is required to be allocated for Host Memory Cache Inbound RDMA Read (Q1) objects. 0x6 = 64 bytes
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.20.61 Private Memory Protocol Engine Q1 Max - GLHMC_PEQ1MAX (0x00522054; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEQ1MAX	27:0	0x8000000	RO	N/A	**Private Memory Protocol Engine Q1 Max** Reports that maximum number of Inbound RDMA Read Queue (Q1) entries supported by the Host Memory Cache.
RESERVED	31:28	0x0	RSV	N/A	Reserved.

13.2.2.20.62 Private Memory Protocol Engine Q1 Free List Max - GLHMC_PEQ1FLMAX (0x00522058; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEQ1FLMAX	25:0	0x2000000	RO	N/A	**Private Memory Protocol Engine Q1 Free List Max** Reports that maximum number of Inbound RDMA Read Queue (Q1) Free List entries supported by the Host Memory Cache.
RESERVED	31:26	0x0	RSV	N/A	Reserved.

13.2.2.20.63 Private Memory FSI Multicast Group Object Size - GLHMC_FSIMCOBJSZ (0x0052205C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMFSIMCOBJSZ	3:0	0x6	RO	N/A	**Private Memory FSI Multicast Object Size** Used to calculate the amount of memory that is required to be allocated for Host Memory Cache FSI Multicast Group objects. 0x6 = 64 bytes
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.20.64 Private Memory FSI Multicast Group Max - GLHMC_FSIMCMAX (0x00522060; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMFSIMCMAX	13:0	0x2000	RO	N/A	**Private Memory FSI Multicast Max** Reports that maximum number of FSI Multicast Group entries supported by the Host Memory Cache.
RESERVED	31:14	0x0	RSV	N/A	Reserved.

13.2.2.20.65 Private Memory FSI Address Vector Object Size - GLHMC_FSIAVOBJSZ (0x00522064; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMFSIAVOBJSZ	3:0	0x6	RO	N/A	**Private Memory FSI Address Vector Object Size** Used to calculate the amount of memory that is required to be allocated for Host Memory Cache FSI Address Vector objects. 0x6 = 64 bytes
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.20.66 Private Memory FSI Address Vector Max - GLHMC_FSIAVMAX (0x00522068; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMFSIAVMAX	17:0	0x20000	RO	N/A	**Private Memory FSI Address Vector Max** Reports that maximum number of FSI Address Vectors supported by the Host Memory Cache.
RESERVED	31:18	0x0	RSV	N/A	Reserved.

13.2.2.20.67 Private Memory Protocol Engine Physical Buffer List Max - GLHMC_PEPBLMAX (0x0052206C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEPBLMAX	28:0	0x10000000	RO	N/A	**Private Memory Protocol Engine Physical Buffer List Max** Reports that maximum number of Physical Buffer List entries supported by the Host Memory Cache.
RESERVED	31:29	0x0	RSV	N/A	Reserved.

13.2.2.20.68 Private Memory Space Segment Descriptor Data Low - GLHMC_FWSDDATALOW (0x00522074; RO)

This register is used in conjunction with PFHMC_SDCMD and PFHMC_SDDATAHIGH to access the Host Memory Cache's segment table.

Field definitions are the same as those defined in Section 13.2.2.20.36.

13.2.2.20.69 Private Memory Space Segment Descriptor Data High - GLHMC_FWSDDATAHIGH (0x00522078; RO)

This register is used in conjunction with PFHMC_SDCMD and PFHMC_SDDATALOW to access the Host Memory Cache's segment table.

Field definitions are the same as those defined in Section 13.2.2.20.37.

13.2.2.20.70 Private Memory Space Page Descriptor Invalidate - GLHMC_FWPDINV (0x0052207C; RO)

This register is used to invalidate cached HMC Page Descriptors that have been set to the invalid state by software.

Field definitions are the same as those defined in Section 13.2.2.20.38.

13.2.2.20.71 Private Memory PE Timer Object Size - GLHMC_PETIMEROBJSZ (0x00522080; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPETIMEROBJSZ	3:0	0x6	RO	N/A	**Private Memory Protocol Engine Timer Object Size** Used to calculate the amount of memory that is required to be allocated for Host Memory Cache Protocol Engine Timer objects. 0x6 = 64 bytes
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.20.72 Private Memory PE Timer Object Max - GLHMC_PETIMERMAX (0x00522084; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPETIMERMAX	28:0	0x10000000	RO	N/A	**Private Memory Protocol Engine Timer Max** Reports that maximum number of PE Timer objects supported by the Host Memory Cache.
RESERVED	31:29	0x0	RSV	N/A	Reserved.

13.2.2.20.73 Private Memory Protocol Engine Read Response Entry Object Size - GLHMC_PERRFOBJSZ (0x00522098; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPERRFOBJSZ	3:0	0x5	RO	N/A	**Private Memory Protocol Engine Read Response FIFO Object Size** Used to calculate the amount of memory that is required to be allocated for Host Memory Cache Read Response FIFO objects. 0x5 = 32 bytes
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.20.74 Private Memory Protocol Engine Read Response FIFO Entry Max - GLHMC_PERRFMAX (0x0052209C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPERRFMAX	27:0	0x8000000	RO	N/A	**Private Memory Protocol Engine Read Response FIFO Max** Reports that maximum number of Read Response FIFO entries supported by the Host Memory Cache.
RESERVED	31:28	0x0	RSV	N/A	Reserved.

13.2.2.20.75 Private Memory Protocol Engine Read Response FIFO Free List Max - GLHMC_PERRFFLMAX (0x005220A0; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPERRFFLMAX	25:0	0x2000000	RO	N/A	**Private Memory Protocol Engine Read Response FIFO Free List Max** Reports that maximum number of Read Response FIFO Free List entries supported by the Host Memory Cache.
RESERVED	31:26	0x0	RSV	N/A	Reserved.

13.2.2.20.76 Private Memory Protocol Engine Out of Order Send Completion (OOISC) FIFO Free List Max - GLHMC_PEOOISCFFLMAX (0x005220A4; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PMPEOOISCFFLMAX	18:0	0x40000	RO	N/A	**Private Memory Protocol Engine Out-of-Order Send Completion FIFO Free List Max**
RESERVED	31:19	0x0	RSV	N/A	Reserved.

13.2.2.20.77 Private Memory Protocol Engine Queue Pair Max - GLHMC_DBQPMAX (0x005220EC; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GLHMC_DBQPMAX	18:0	0x40000	RO	N/A	**Doorbell QP Max** Reports the maximum number of Protocol Engine Queue Pairs supported by the Host Memory Cache and Doorbell array.
RESERVED	31:19	0x0	RSV	N/A	Reserved.

13.2.2.20.78 Private Memory Protocol Engine Completion Queue Max - GLHMC_DBCQMAX (0x005220F0; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GLHMC_DBCQMAX	19:0	0x080000	RO	N/A	**Doorbell CQ Max** Reports the maximum number of Protocol Engine Completion Queues supported by the Host Memory Cache and Doorbell array.
RESERVED	31:20	0x0	RSV	N/A	Reserved.

13.2.2.20.79 FPM PE QP Base - GLHMC_PEQPBASE[n] (0x00524000 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPEQPBASE	23:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Engine QP Base** Reports the Function Private Memory space base address for the Protocol Engine Queue Pair objects in 512-byte increments. In other words, the value is this registers must be multiplied by 512 to get the actual address out of the 8 GB FPM address space. This register is updated by hardware when the Commit FPM Values CQP operation is performed. Other than for debug purposes, this register should be treated as Read Only.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.80 FPM PE QP Object Count - GLHMC_PEQPCNT[n] (0x00524100 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPEQPCNT	28:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Engine QP Count** Used to set the Function Private Memory space size for the Protocol Engine Queue Pair objects. The associated base register is updated after Commit FPM Values CQP operation is performed to indicate to hardware that all of the FPM size registers have been set properly and the FPM map should be recomputed.
RESERVED	31:29	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.81 FPM PE CQ Base - GLHMC_PECQBASE[n] (0x00524200 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPECQBASE	23:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Engine CQ Base** Reports the Function Private Memory space base address for the Protocol Engine Completion Queue objects in 512-byte increments. In other words, the value is this registers must be multiplied by 512 to get the actual address out of the 8 GB FPM address space. This register is updated by hardware when the Commit FPM Values CQP operation is performed. Other than for debug purposes, this register should be treated as Read Only.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.82 FPM PE CQ Object Count - GLHMC_PECQCNT[n] (0x00524300 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPECQCNT	28:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Engine CQ Count** Used to set the Function Private Memory space size for the Protocol Engine Completion Queue objects. The associated base register is updated after the Commit FPM Values CQP operation is performed to indicate to hardware that all of the FPM size registers have been set properly and the FPM map should be recomputed.
RESERVED	31:29	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.83 FPM PE Hash Table Entry Base - GLHMC_PEHTEBASE[n] (0x00524600 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPEHTEBASE	23:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Engine Hash Table Entry Base** Reports the Function Private Memory space base address for the Protocol Engine Hash Table Entry objects in 512-byte increments. In other words, the value is this registers must be multiplied by 512 to get the actual address out of the 8 GB FPM address space. This register is updated by hardware when the Commit FPM Values CQP operation is performed. Other than for debug purposes, this register should be treated as Read Only.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.84 FPM PE Hash Table Object Count - GLHMC_PEHTCNT[n] (0x00524700 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPEHTCNT	28:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Engine Hash Table Count** Used to set the Function Private Memory space size for the Protocol Engine Hash Table objects. The associated base register is updated after the Commit FPM Values CQP operation is performed to indicate to hardware that all of the FPM size registers have been set properly and the FPM map should be recomputed.
RESERVED	31:29	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.85 FPM PE ARP Table Base - GLHMC_PEARPBASE[n] (0x00524800 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPEARPBASE	23:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Engine ARP Base** Reports the Function Private Memory space base address for the Protocol Engine ARP Table objects in 512-byte increments. In other words, the value is this registers must be multiplied by 512 to get the actual address out of the 8 GB FPM address space. This register is updated by hardware when the Commit FPM Values CQP operation is performed. Other than for debug purposes, this register should be treated as Read Only.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.86 FPM PE ARP Table Object Count - GLHMC_PEARPCNT[n] (0x00524900 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPEARPCNT	28:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Engine ARP Count** Used to set the Function Private Memory space size for the Protocol Engine ARP table objects. The associated base register is updated after the Commit FPM Values CQP operation is performed to indicate to hardware that all of the FPM size registers have been set properly and the FPM map should be recomputed.
RESERVED	31:29	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.87 FPM PE APBVT In-Use Base - GLHMC_APBVTINUSEBASE[n] (0x00524A00 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMAPBINUSEBASE	23:0	0x0	RW	UNDEFINED	**Function Private Memory APBVT In-Use Base** Reports the base Function Private Memory Space Address of the Accelerated Port Bit In-Use object in 512-byte increments. There is a single APBVT In-Use object for each PCI function and it is a fixed 8 KB in size. Since there is only a single object and it is a fixed size, there is not a corresponding Object Size or region size register. This register is updated by hardware when the Commit FPM Values CQP operation is performed.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.88 FPM PE MRT Base - GLHMC_PEMRBASE[n] (0x00524C00 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPEMRBASE	23:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Engine Memory Region Base** Reports the Function Private Memory space base address for the Protocol Engine Memory Region Table objects in 512-byte increments. In other words, the value is this registers must be multiplied by 512 to get the actual address out of the 8 GB FPM address space. This register is updated by hardware when the Commit FPM Values CQP operation is performed. Other than for debug purposes, this register should be treated as Read Only.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.89 FPM PE Memory Region Table Object Count - GLHMC_PEMRCNT[n] (0x00524D00 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPEMRSZ	28:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Engine Memory Region Size** Used to set the Function Private Memory space size for the Protocol Engine Memory Region Table objects. The associated base register is updated after the Commit FPM Values CQP operation is performed to indicate to hardware that all of the FPM size registers have been set properly and the FPM map should be recomputed.
RESERVED	31:29	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.90 FPM PE Xmit FIFO Base - GLHMC_PEXFBASE[n] (0x00524E00 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPEXFBASE	23:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Transmit FIFO Base** Reports the Function Private Memory space base address for the Protocol Engine Xmit FIFO objects in 512-byte increments. In other words, the value is this registers must be multiplied by 512 to get the actual address out of the 8 GB FPM address space. This register is updated by hardware when the Commit FPM Values CQP operation is performed. Other than for debug purposes, this register should be treated as Read Only.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.91 FPM PE Xmit FIFO Object Count - GLHMC_PEXFCNT[n] (0x00524F00 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPEXFCNT	28:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Transmit FIFO Count** Used to set the Function Private Memory space size for the Protocol Engine Xmit FIFO objects. The associated base register is updated after the Commit FPM Values CQP operation is performed to indicate to hardware that all of the FPM size registers have been set properly and the FPM map should be recomputed.
RESERVED	31:29	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.92 FPM PE Xmit FIFO Free List Base - GLHMC_PEXFFLBASE[n] (0x00525000 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPEXFFLBASE	23:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Transmit FIFO Free List Base** Reports the Function Private Memory space base address for the Protocol Engine Xmit FIFO Free List objects in 512-byte increments. In other words, the value is this registers must be multiplied by 512 to get the actual address out of the 8 GB FPM address space. This register is updated by hardware when the Commit FPM Values CQP operation is performed. Other than for debug purposes, this register should be treated as Read Only.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.93 FPM PE IRRQ Base - GLHMC_PEQ1BASE[n] (0x00525200 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPEQ1BASE	23:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Engine Q1 Base** Reports the Function Private Memory space base address for the Protocol Engine Inbound RDMA Read Queue (Q1) objects in 512-byte increments. In other words, the value is this registers must be multiplied by 512 to get the actual address out of the 8 GB FPM address space. This register is updated by hardware when the Commit FPM Values CQP operation is performed. Other than for debug purposes, this register should be treated as Read Only.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.94 FPM PE IRRQ Object Count - GLHMC_PEQ1CNT[n] (0x00525300 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPEQ1CNT	28:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Engine Q1 Count** Used to set the Function Private Memory space size for the Protocol Engine Inbound RDMA Read Queue (Q1) objects. The associated base register is updated after the Commit FPM Values CQP operation is performed to indicate to hardware that all of the FPM size registers have been set properly and the FPM map should be recomputed.
RESERVED	31:29	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.95 FPM PE IRRQ Free List Base - GLHMC_PEQ1FLBASE[n] (0x00525400 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPEQ1FLBASE	23:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Engine Q1 Free List Base** Reports the Function Private Memory space base address for the Protocol Engine Inbound RDMA Read Queue (Q1) Free List objects in 512-byte increments. In other words, the value is this registers must be multiplied by 512 to get the actual address out of the 8 GB FPM address space. This register is updated by hardware when the Commit FPM Values CQP operation is performed. Other than for debug purposes, this register should be treated as Read Only.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.96 FPM FSI Address Vector Base - GLHMC_FSIAVBASE[n] (0x00525600 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMFSIAVBASE	23:0	0x0	RW	UNDEFINED	**Function Private Memory FSI Address Vector Base** Reports the Function Private Memory space base address for the FSI Address Vector objects in 512-byte increments. In other words, the value is this registers must be multiplied by 512 to get the actual address out of the 8 GB FPM address space. This register is updated by hardware when the Commit FPM Values CQP operation is performed. Other than for debug purposes, this register should be treated as Read Only.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.97 FPM FSI Address Vector Object Count - GLHMC_FSIAVCNT[n] (0x00525700 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMFSIAVCNT	28:0	0x0	RW	UNDEFINED	**Function Private Memory FSI Address Vector Count** Used to set the Function Private Memory space size for the FSI Address Vector objects. The associated base register is updated after the Commit FPM Values CQP operation is performed to indicate to hardware that all of the FPM size registers have been set properly and the FPM map should be recomputed.
RESERVED	31:29	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.98 FPM PE Physical Buffer List Base - GLHMC_PEPBLBASE[n] (0x00525800 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPEPBLBASE	23:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Engine Physical Buffer List Base** Reports the Function Private Memory space base address for the Protocol Engine Physical Buffer List objects in 512-byte increments. In other words, the value is this registers must be multiplied by 512 to get the actual address out of the 8 GB FPM address space. This register is updated by hardware when the Commit FPM Values CQP operation is performed. Other than for debug purposes, this register should be treated as Read Only.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.99 FPM PE PBL Object Count - GLHMC_PEPBLCNT[n] (0x00525900 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPEPBLCNT	28:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Engine Physical Buffer List Count** Used to set the Function Private Memory space size for the Protocol Engine Physical Buffer List objects. The associated base register is updated after the Commit FPM Values CQP operation is performed to indicate to hardware that all of the FPM size registers have been set properly and the FPM map should be recomputed.
RESERVED	31:29	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.100 FPM PE Timer Base - GLHMC_PETIMERBASE[n] (0x00525A00 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPETIMERBASE	23:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Engine Timer Base** Reports the Function Private Memory space base address for the Protocol Engine Timer objects in 512-byte increments. In other words, the value of this registers must be multiplied by 512 to get the actual address out of the 8 GB FPM address space. This register is updated by hardware when the Commit FPM Values CQP operation is performed. Other than for debug purposes, this register should be treated as Read Only.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.101 FPM PE Timer Object Count - GLHMC_PETIMERCNT[n] (0x00525B00 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPETIMERCNT	28:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Engine Timer Count** Used to set the Function Private Memory space size for the Protocol Engine Timer objects. The associated base register is updated after the Commit FPM Values CQP operation is performed to indicate to hardware that all of the FPM size registers have been set properly and the FPM map should be recomputed.
RESERVED	31:29	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.102 FPM FSI Multicast Group Base - GLHMC_FSIMCBASE[n] (0x00526000 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMFSIMCBASE	23:0	0x0	RW	UNDEFINED	**Function Private Memory FSI Multicast Base** Reports the Function Private Memory space base address for the FSI Multicast Group objects in 512-byte increments. In other words, the value is this registers must be multiplied by 512 to get the actual address out of the 8 GB FPM address space. This register is updated by hardware when the Commit FPM Values CQP operation is performed. Other than for debug purposes, this register should be treated as Read Only.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.103 FPM FSI Multicast Group Object Count - GLHMC_FSIMCCNT[n] (0x00526100 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMFSIMCSZ	28:0	0x0	RW	UNDEFINED	**Function Private Memory FSI Multicast Size** Used to set the Function Private Memory space size for the FSI Multicast Group objects. The associated base register is updated after the Commit FPM Values CQP operation is performed to indicate to hardware that all of the FPM size registers have been set properly and the FPM map should be recomputed.
RESERVED	31:29	0x0	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.104 FPM PE Header Base - GLHMC_PEHDRBASE[n] (0x00526200 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GLHMC_PEHDRBASE	31:0	0x0	RW	UNDEFINED	**Protocol Engine Header Base** Reports the Function Private Memory space base address for the Protocol Engine Header objects in 512-byte increments. In other words, the value in this register must be multiplied by 512 to get the actual address out of the 8 GB FPM address space. This register is updated by hardware when the Commit FPM Values CQP operation is performed. Other than for debug purposes, this register should be treated as Read Only.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.105 FPM PE Header Object Count - GLHMC_PEHDRCNT[n] (0x00526300 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GLHMC_PEHDRCNT	31:0	0x0	RW	UNDEFINED	**Protocol Engine Header Count** Used to set the Function Private Memory space size for the Protocol Engine Header objects. The associated base register is updated after the Commit FPM Values CQP operation is performed to indicate to hardware that all of the FPM size registers have been set properly and the FPM map should be recomputed.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.106 FPM PE Metadata Base - GLHMC_PEMDBASE[n] (0x00526400 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GLHMC_PEMDBASE	31:0	0x0	RW	UNDEFINED	**Protocol Engine Metadata Base** Reports the Function Private Memory space base address for the Protocol Engine Metadata objects in 512-byte increments. In other words, the value in this register must be multiplied by 512 to get the actual address out of the 8 GB FPM address space. This register is updated by hardware when the Commit FPM Values CQP operation is performed. Other than for debug purposes, this register should be treated as Read Only.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.107 FPM PE Metadata Object Count - GLHMC_PEMDCNT[n] (0x00526500 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GLHMC_PEMDCNT	31:0	0x0	RW	UNDEFINED	**Protocol Engine Metadata Count** Used to set the Function Private Memory space size for the Protocol Engine Metadata objects. The associated base register is updated after the Commit FPM Values CQP operation is performed to indicate to hardware that all of the FPM size registers have been set properly and the FPM map should be recomputed.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.108 FPM PE Out of Order Send Completion Base - GLHMC_PEOOISCBASE[n] (0x00526600 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GLHMC_PEOOISCBASE	31:0	0x0	RW	UNDEFINED	**Protocol Engine Out-of-Order Send Completion Base** Reports the Function Private Memory space base address for the Protocol Engine Out of Order Send Completion objects in 512-byte increments. In other words, the value in this register must be multiplied by 512 to get the actual address out of the 8 GB FPM address space. This register is updated by hardware when the Commit FPM Values CQP operation is performed. Other than for debug purposes, this register should be treated as Read Only.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.109 FPM PE Out of Order Send Completion Object Count - GLHMC_PEOOISCCNT[n] (0x00526700 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GLHMC_PEOOISCCNT	31:0	0x0	RW	UNDEFINED	**Protocol Engine Out-of-Order Send Completion Count** Used to set the Function Private Memory space size for the Protocol Engine Out of Order Send Completion objects. The associated base register is updated after Commit FPM Values CQP operation is performed to indicate to hardware that all of the FPM size registers have been set properly and the FPM map should be recomputed.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.110 FPM PE Read Response Base - GLHMC_PERRFBASE[n] (0x00526800 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GLHMC_PERRFBASE	31:0	0x0	RW	UNDEFINED	**Protocol Engine Read Response FIFO Base** Reports the Function Private Memory space base address for the Protocol Engine Read Response objects in 512-byte increments. In other words, the value in this register must be multiplied by 512 to get the actual address out of the 8 GB FPM address space. This register is updated by hardware when the Commit FPM Values CQP operation is performed. Other than for debug purposes, this register should be treated as Read Only.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.111 FPM PE Read Response Object Count - GLHMC_PERRFCNT[n] (0x00526900 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GLHMC_PERRFCNT	31:0	0x0	RW	UNDEFINED	**Protocol Engine Read Response FIFO Count** Used to set the Function Private Memory space size for the Protocol Engine Read Response objects. The associated base register is updated after the Commit FPM Values CQP operation is performed to indicate to hardware that all of the FPM size registers have been set properly and the FPM map should be recomputed.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.112 FPM PE Read Response FIFO Free List Base - GLHMC_PERRFFLBASE[n] (0x00526A00 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GLHMC_PERRFFLBASE	31:0	0x0	RW	UNDEFINED	**Protocol Engine Read Response FIFO Free List Base** Reports the Function Private Memory space base address for the Protocol Engine Read Response FIFO Free List objects in 512-byte increments. In other words, the value in this register must be multiplied by 512 to get the actual address out of the 8 GB FPM address space. This register is updated by hardware when the Commit FPM Values CQP operation is performed. Other than for debug purposes, this register should be treated as Read Only.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.113 FPM PE Read Response FIFO Free List Object Count - GLHMC_PERRFFLCNT_PMAT[n] (0x00526B00 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPERRFFLCNT	28:0	0x0	RW	UNDEFINED	**Function Private Memory Protocol Engine Read Response FIFO Free List Count** Used to set the Function Private Memory space size for the Protocol Engine Read Response FIFO objects. The associated base register is updated after the Commit FPM Values CQP operation is performed to indicate to hardware that all of the FPM size registers have been set properly and the FPM map should be recomputed.
RESERVED	31:29	000b	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.114 FPM PE Out of Order Send Completion (OOISC) FIFO Free List Base - GLHMC_PEOOISCFFLBASE[n] (0x00526C00 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GLHMC_PEOOISCFFLBASE	31:0	0x0	RW	UNDEFINED	**Protocol Engine Out-of-Order Send Completion FIFO Free List Base** Reports the Function Private Memory space base address for the Protocol Engine Out of Order Send Completion (OOISC) FIFO Free List objects in 512-byte increments. In other words, the value in this register must be multiplied by 512 to get the actual address out of the 8 GB FPM address space. This register is updated by hardware when the Commit FPM Values CQP operation is performed. Other than for debug purposes, this register should be treated as Read Only.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.115 FPM PE Out of Order Send Completion (OOISC) FIFO Free List Object Count - GLHMC_PEOOISCFFLCNT_PMAT[n] (0x00526D00 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FPMPEOOISCFLCNT	28:0	0x0	RW	UNDEFINED	**Protocol Engine Out-of-Order Send Completion Free List Count** Used to set the Function Private Memory space size for the Protocol Engine Out of Order Send Completion (OOISC) FIFO objects. The associated base register is updated after the Commit FPM Values CQP operation is performed to indicate to hardware that all of the FPM size registers have been set properly and the FPM map should be recomputed.
RESERVED	31:29	000b	RSV	N/A	Reserved.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.116 Private Memory Space VF Segment Descriptor Data Low - GLHMC_VFSDDATALOW[n] (0x00528100 + 0x4*n, n=0...31; RO)

This register is used in conjunction with PFHMC_SDCMD and PFHMC_SDDATAHIGH to access the Host Memory Cache's segment table.

Field definitions are the same as those defined in Section 13.2.2.20.36.

Note: 16 instances of this register are implemented for this product. The remaining instances are reserved for future expansion

13.2.2.20.117 Private Memory Space VF Segment Descriptor Data High - GLHMC_VFSDDATAHIGH[n] (0x00528200 + 0x4*n, n=0...31; RO)

This register is used in conjunction with PFHMC_SDCMD and PFHMC_SDDATALOW to access the Host Memory Cache's segment table.

Field definitions are the same as those defined in Section 13.2.2.20.37.

Note: 16 instances of this register are implemented for this product. The remaining instances are reserved for future expansion

13.2.2.20.118 Private Memory Space Page Descriptor Invalidate - GLHMC_VFPDINV[n] (0x00528300 + 0x4*n, n=0...31; RO)

This register is used to invalidate cached HMC Page Descriptors that have been set to the invalid state by software.

Field definitions are the same as those defined in Section 13.2.2.20.38.

Note: 16 instances of this register are implemented for this product. The remaining instances are reserved for future expansion

13.2.2.20.119 Private Memory Segment Table Partitioning Registers - GLHMC_VFSDPART[n] (0x00528800 + 0x4*n, n=0...31; RO)

This register is used to partition the shared Host Memory Cache segment table.

Field definitions are the same as those defined in Section 13.2.2.20.41.

Note: This register must be read only in the PF CSR Space.

13.2.2.20.120 FPM PE QP Base - GLHMC_VFPEQPBASE[n] (0x0052C000 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.79.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.121 FPM PE QP Object Count - GLHMC_VFPEQPCNT[n] (0x0052C100 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.80.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.122 FPM PE CQ Base - GLHMC_VFPECQBASE[n] (0x0052C200 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.81.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.123 FPM PE CQ Object Count - GLHMC_VFPECQCNT[n] (0x0052C300 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.82.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.124 FPM PE Hash Table Entry Base - GLHMC_VFPEHTEBASE[n] (0x0052C600 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.83.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.125 FPM PE Hash Table Object Count - GLHMC_VFPEHTCNT[n] (0x0052C700 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.84.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.126 FPM PE ARP Table Base - GLHMC_VFPEARPBASE[n] (0x0052C800 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.85.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.127 FPM PE ARP Table Object Count - GLHMC_VFPEARPCNT[n] (0x0052C900 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.86.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.128 FPM PE APBVT In-Use Base - GLHMC_VFAPBVTINUSEBASE[n] (0x0052CA00 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.87.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.129 FPM PE MRT Base - GLHMC_VFPEMRBASE[n] (0x0052CC00 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.88.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.130 FPM PE Memory Region Table Object Count - GLHMC_VFPEMRCNT[n] (0x0052CD00 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.89.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.131 FPM PE Xmit FIFO Base - GLHMC_VFPEXFBASE[n] (0x0052CE00 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.90.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.132 FPM PE Xmit FIFO Object Count - GLHMC_VFPEXFCNT[n] (0x0052CF00 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.91.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.133 FPM PE Xmit FIFO Free List Base - GLHMC_VFPEXFFLBASE[n] (0x0052D000 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.92.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.134 FPM PE IRRQ Base - GLHMC_VFPEQ1BASE[n] (0x0052D200 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.93.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.135 FPM PE IRRQ Object Count - GLHMC_VFPEQ1CNT[n] (0x0052D300 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.94.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.136 FPM PE IRRQ Free List Base - GLHMC_VFPEQ1FLBASE[n] (0x0052D400 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.95.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.137 FPM FSI Address Vector Base - GLHMC_VFFSIAVBASE[n] (0x0052D600 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.96.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.138 FPM FSI Address Vector Object Count - GLHMC_VFFSIAVCNT[n] (0x0052D700 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.97.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.139 FPM PE Physical Buffer List Base - GLHMC_VFPEPBLBASE[n] (0x0052D800 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.98.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.140 FPM PE PBL Object Count - GLHMC_VFPEPBLCNT[n] (0x0052D900 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.99.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.141 FPM PE Timer Base - GLHMC_VFPETIMERBASE[n] (0x0052DA00 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.100.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.142 FPM PE Timer Object Count - GLHMC_VFPETIMERCNT[n] (0x0052DB00 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.101.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.143 FPM FSI Multicast Group Base - GLHMC_VFFSIMCBASE[n] (0x0052E000 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.102.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.144 FPM FSI Multicast Group Object Count - GLHMC_VFFSIMCCNT[n] (0x0052E100 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.103.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.145 FPM PE Header Base - GLHMC_VFPEHDRBASE[n] (0x0052E200 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.104.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.146 FPM PE Header Object Count - GLHMC_VFPEHDRCNT[n] (0x0052E300 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.105.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.147 FPM PE Metadata Base - GLHMC_VFPEMDBASE[n] (0x0052E400 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.106.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.148 FPM PE Metadata Object Count - GLHMC_VFPEMDCNT[n] (0x0052E500 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.107.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.149 FPM PE Out of Order Send Completion Base - GLHMC_VFPEOOISCBASE[n] (0x0052E600 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.108.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.150 FPM PE Out of Order Send Completion Object Count - GLHMC_VFPEOOISCCNT[n] (0x0052E700 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.109.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.151 FPM PE Read Response Base - GLHMC_VFPERRFBASE[n] (0x0052E800 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.110.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.152 FPM PE Read Response Object Count - GLHMC_VFPERRFCNT[n] (0x0052E900 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.111.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.153 FPM PE Read Response FIFO Free List Base - GLHMC_VFPERRFFLBASE[n] (0x0052EA00 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.112.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.154 FPM PE Out of Order Send Completion (OOISC) FIFO Free List Base - GLHMC_VFPEOOISCFFLBASE[n] (0x0052EC00 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.20.114.

Note: Eight instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.20.155 PDOC Cache Attributes - GLPDOC_CACHESIZE (0x00530048; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
WORD_SIZE	7:0	0x40	RO	N/A	**Word Size** The cache line size in bytes.
SETS	19:8	0x40	RO	N/A	**Sets** The number of cache sets.
WAYS	23:20	0x8	RO	N/A	**Ways** The number of cache ways.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.21 PF - Context Manager Registers

13.2.2.21.1 CMPE Cache Attributes - GLCM_PE_CACHESIZE (0x005046B4; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
WORD_SIZE	11:0	0x200	RO	N/A	**Word Size** The cache line size in bytes.
SETS	15:12	0x1	RO	N/A	**Sets** The number of cache sets.
WAYS	24:16	0x100	RO	N/A	**Ways** The number of cache ways.
RESERVED	31:25	0x0	RSV	N/A	Reserved.

13.2.2.22 PF - Control Queues Registers

Registers related to the various types of control queues: firmware Admin Queues, sideband queues, and mailbox queues.

13.2.2.22.1 PF Firmware Admin Transmit Queue Base Address Low - PF_FW_ATQBAL (0x00080000; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ATQBAL_LSB	5:0	0x0	RO	N/A	**Admin Transmit Queue Base Address Low LSB** Tied to zero to achieve alignment.
ATQBAL	31:6	0x0	RW	UNDEFINED	**Admin Transmit Queue Base Address Low** Transmit descriptor base address low. Must be 64-byte aligned (together with ATQBAL_LSB).

13.2.2.22.2 Global Tools Firmware Admin Transmit Queue Base Address Low - GL_FW_TOOL_ATQBAL (0x00080040; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.1.

13.2.2.22.3 PF0 PSM Firmware Admin Transmit Queue Base Address Low - PF0_FW_PSM_ATQBAL (0x00080044; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.1.

13.2.2.22.4 PF0 HLP Firmware Admin Transmit Queue Base Address Low - PF0_FW_HLP_ATQBAL (0x00080048; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.1.

13.2.2.22.5 PF Firmware Admin Receive Queue Base Address Low - PF_FW_ARQBAL (0x00080080; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ARQBAL_LSB	5:0	0x0	RO	N/A	**Admin Receive Queue Base Address Low LSB** Tied to zero to achieve alignment.
ARQBAL	31:6	0x0	RW	UNDEFINED	**Admin Receive Queue Base Address Low** Receive descriptor base address low. Must be 64-byte aligned (together with ARQBAL_LSB).

13.2.2.22.6 Global Tools Firmware Admin Receive Queue Base Address Low - GL_FW_TOOL_ARQBAL (0x000800C0; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.5.

13.2.2.22.7 PF0 PSM Firmware Admin Receive Queue Base Address Low - PF0_FW_PSM_ARQBAL (0x000800C4; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.5.

13.2.2.22.8 PF0 HLP Firmware Admin Receive Queue Base Address Low - PF0_FW_HLP_ARQBAL (0x000800C8; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.5.

13.2.2.22.9 PF Firmware Admin Transmit Queue Base Address High - PF_FW_ATQBAH (0x00080100; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ATQBAH	31:0	0x0	RW	UNDEFINED	**Admin Transmit Queue Base Address High** Transmit descriptor base address high.

13.2.2.22.10 Global Tools Firmware Admin Transmit Queue Base Address High - GL_FW_TOOL_ATQBAH (0x00080140; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.9.

13.2.2.22.11 PF0 PSM Firmware Admin Transmit Queue Base Address High - PF0_FW_PSM_ATQBAH (0x00080144; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.9.

13.2.2.22.12 PF0 HLP Firmware Admin Transmit Queue Base Address High - PF0_FW_HLP_ATQBAH (0x00080148; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.9.

13.2.2.22.13 PF Firmware Admin Receive Queue Base Address High - PF_FW_ARQBAH (0x00080180; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ARQBAH	31:0	0x0	RW	UNDEFINED	**Admin Receive Queue Base Address High** Receive descriptor base address high.

13.2.2.22.14 Global Tools Firmware Admin Receive Queue Base Address High - GL_FW_TOOL_ARQBAH (0x000801C0; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.13.

13.2.2.22.15 PF0 PSM Firmware Admin Receive Queue Base Address High - PF0_FW_PSM_ARQBAH (0x000801C4; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.13.

13.2.2.22.16 PF0 HLP Firmware Admin Receive Queue Base Address High - PF0_FW_HLP_ARQBAH (0x000801C8; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.13.

13.2.2.22.17 PF Firmware Admin Transmit Queue Length - PF_FW_ATQLEN (0x00080200; RW)

This register sets the size of the ring. Maximum size is 1024.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ATQLEN	9:0	0x0	RW	UNDEFINED	**Admin Transmit Queue Length** Descriptor ring length. Max size is 1024.
RESERVED	27:10	0x0	RSV	N/A	Reserved.
ATQVFE	28	0b	RW	UNDEFINED	**Admin Transmit Queue VF Error** VF error. This bit is set by firmware on a PF queue when one of its VFs had an Admin Queue error.
ATQOVFL	29	0b	RW	UNDEFINED	**Admin Transmit Queue Overflow** Overflow error. This bit is set by firmware when a message was lost because there was no room on the queue.
ATQCRIT	30	0b	RW	UNDEFINED	**Admin Transmit Queue Critical** Critical error. This bit is set by firmware when a critical error has been detected on this queue.
ATQENABLE	31	0b	RW	UNDEFINED	**Admin Transmit Queue Enable** Enable bit. This bit is set by driver to indicate that the queue is active. When setting the enable bit, software should initialize all other fields. This flag is implemented by a FF and cleared by PFR.

13.2.2.22.18 Global Tools Firmware Admin Transmit Queue Length - GL_FW_TOOL_ATQLEN (0x00080240; RW)

This register sets the size of the ring. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.17.

13.2.2.22.19 PF0 PSM Firmware Admin Transmit Queue Length - PF0_FW_PSM_ATQLEN (0x00080244; RW)

This register sets the size of the ring. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.17.

13.2.2.22.20 PF0 HLP Firmware Admin Transmit Queue Length - PF0_FW_HLP_ATQLEN (0x00080248; RW)

This register sets the size of the ring. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.17.

13.2.2.22.21 PF Firmware Admin Receive Queue Length - PF_FW_ARQLEN (0x00080280; RW)

This register specifies the receive queue length. Maximum size is 1024.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ARQLEN	9:0	0x0	RW	UNDEFINED	**Admin Receive Queue Length** Descriptor ring length. Max size is 1024.
RESERVED	27:10	0x0	RSV	N/A	Reserved.
ARQVFE	28	0b	RW	UNDEFINED	**Admin Receive Queue VF Error** VF error. This bit is set by firmware on a PF queue when one of its VFs had an Admin Queue error.
ARQOVFL	29	0b	RW	UNDEFINED	**Admin Receive Queue Overflow** Overflow error. This bit is set by firmware when a message was lost because there was no room on the queue.
ARQCRIT	30	0b	RW	UNDEFINED	**Admin Receive Queue Critical** Critical error. This bit is set by firmware when a critical error has been detected on this queue.
ARQENABLE	31	0b	RW	UNDEFINED	**Admin Receive Queue Enable** Enable bit. This bit is set by driver to indicate that the queue is active. When setting the enable bit, software should initialize all other fields. This flag is implemented by a FF and cleared by PFR.

13.2.2.22.22 Global Tools Firmware Admin Receive Queue Length - GL_FW_TOOL_ARQLEN (0x000802C0; RW)

This register specifies the receive queue length. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.21.

13.2.2.22.23 PF0 PSM Firmware Admin Receive Queue Length - PF0_FW_PSM_ARQLEN (0x000802C4; RW)

This register specifies the receive queue length. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.21.

13.2.2.22.24 PF0 HLP Firmware Admin Receive Queue Length - PF0_FW_HLP_ARQLEN (0x000802C8; RW)

This register specifies the receive queue length. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.21.

13.2.2.22.25 PF Firmware Admin Transmit Head - PF_FW_ATQH (0x00080300; RW)

Admin transmit queue head pointer.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ATQH	9:0	0x0	RW	UNDEFINED	**Admin Transmit Queue Head** Transmit queue head pointer. At queue initialization, the software clears the head pointer. During nominal operation, the firmware increments the head following command execution.
RESERVED	31:10	0x0	RSV	N/A	Reserved.

13.2.2.22.26 Global Tools Firmware Admin Transmit Head - GL_FW_TOOL_ATQH (0x00080340; RW)

Admin transmit queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.25.

13.2.2.22.27 PF0 PSM Firmware Admin Transmit Head - PF0_FW_PSM_ATQH (0x00080344; RW)

Admin transmit queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.25.

13.2.2.22.28 PF0 HLP Firmware Admin Transmit Head - PF0_FW_HLP_ATQH (0x00080348; RW)

Admin transmit queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.25.

13.2.2.22.29 PF Firmware Admin Receive Queue Head - PF_FW_ARQH (0x00080380; RW)

Admin receive queue head pointer.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ARQH	9:0	0x0	RW	UNDEFINED	**Admin Receive Queue Head** Receive queue head pointer. At queue initialization, the software clears the head pointer. During nominal operation, the firmware increments the head following command execution.
RESERVED	31:10	0x0	RSV	N/A	Reserved.

13.2.2.22.30 Global Tools Firmware Admin Receive Queue Head - GL_FW_TOOL_ARQH (0x000803C0; RW)

Admin receive queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.29.

13.2.2.22.31 PF0 PSM Firmware Admin Receive Queue Head - PF0_FW_PSM_ARQH (0x000803C4; RW)

Admin receive queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.29.

13.2.2.22.32 PF0 HLP Firmware Admin Receive Queue Head - PF0_FW_HLP_ARQH (0x000803C8; RW)

Admin receive queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.29.

13.2.2.22.33 PF Firmware Admin Transmit Tail - PF_FW_ATQT (0x00080400; RW)

Admin transmit queue tail pointer.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ATQT	9:0	0x0	RW	UNDEFINED	**Admin Transmit Queue Tail** Transmit queue tail pointer. Incremented to indicate that there are new valid descriptors on the ring. Software can only write to this register once both transmit and receive queues are properly initialized. And clear to zero at queue initialization.
RESERVED	31:10	0x0	RSV	N/A	Reserved.

13.2.2.22.34 Global Tools Firmware Admin Transmit Tail - GL_FW_TOOL_ATQT (0x00080440; RW)

Admin transmit queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.33.

13.2.2.22.35 PF0 PSM Firmware Admin Transmit Tail - PF0_FW_PSM_ATQT (0x00080444; RW)

Admin transmit queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.33.

13.2.2.22.36 PF0 HLP Firmware Admin Transmit Tail - PF0_FW_HLP_ATQT (0x00080448; RW)

Admin transmit queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.33.

13.2.2.22.37 PF Firmware Admin Receive Queue Tail - PF_FW_ARQT (0x00080480; RW)

Admin receive queue tail pointer.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ARQT	9:0	0x0	RW	UNDEFINED	**Admin Receive Queue Tail** Receive queue tail pointer. Incremented to indicate that there are new valid descriptors on the ring. Software can only write to this register once the queue is fully configured. And clear to zero at queue initialization.
RESERVED	31:10	0x0	RSV	N/A	Reserved.

13.2.2.22.38 Global Tools Firmware Admin Receive Queue Tail - GL_FW_TOOL_ARQT (0x000804C0; RW)

Admin receive queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.37.

13.2.2.22.39 PF0 PSM Firmware Admin Receive Queue Tail - PF0_FW_PSM_ARQT (0x000804C4; RW)

Admin receive queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.37.

13.2.2.22.40 PF0 HLP Firmware Admin Receive Queue Tail - PF0_FW_HLP_ARQT (0x000804C8; RW)

Admin receive queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.37.

13.2.2.22.41 Tools Mailbox HOST Interface Buffer Area - GL_HIBA[n] (0x00081000 + 0x4*n, n=0...1023; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GL_HIBA	31:0	0x0	RW	UNDEFINED	**Host Interface Buffer Area**

13.2.2.22.42 Tools Mailbox HOST Interface Descriptor Area - GL_HIDA[n] (0x00082000 + 0x4*n, n=0...15; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GL_HIDA	31:0	0x0	RW	UNDEFINED	**Host Interface Descriptor Area**

13.2.2.22.43 Tools Mailbox HOST Interface Control Register - GL_HICR (0x00082040; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	0	0b	RSV	N/A	Reserved.
C	1	0b	RW	UNDEFINED	**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.
SV	2	0b	RW	UNDEFINED	**Status Valid** Indicates that there is a valid status in the first descriptor that the device driver can read. If Flags.BUF is set in the first descriptor, the buffer is valid. 0b = Status not valid. 1b = Status valid.
EV	3	0b	RW	UNDEFINED	**Event Valid** Indicates that there is a valid status in the second descriptor that the device driver can read. If Flags.BUF is set in the second descriptor, the buffer is valid. 0b = Status not valid. 1b = Status valid.
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.22.44 Tools Mailbox HOST Interface Enable Register - GL_HICR_EN (0x00082044; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
EN	0	0b	RO	UNDEFINED	**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	UNDEFINED	Reserved.

13.2.2.22.45 VSI Mailbox Transmit Queue Base Address Low - VSI_MBX_ATQBAL[VSI] (0x00220000 + 0x4*VSI, VSI=0...767; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.55.

13.2.2.22.46 VSI Mailbox Transmit Queue Base Address High - VSI_MBX_ATQBAH[VSI] (0x00221000 + 0x4*VSI, VSI=0...767; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.56.

13.2.2.22.47 VSI Mailbox Transmit Queue Length - VSI_MBX_ATQLEN[VSI] (0x00222000 + 0x4*VSI, VSI=0...767; RW)

This register sets the size of the ring. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.57.

13.2.2.22.48 VSI Mailbox Transmit Head - VSI_MBX_ATQH[VSI] (0x00223000 + 0x4*VSI, VSI=0...767; RW)

Admin transmit queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.58.

13.2.2.22.49 VSI Mailbox Transmit Tail - VSI_MBX_ATQT[VSI] (0x00224000 + 0x4*VSI, VSI=0...767; RW)

Admin transmit queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.59.

13.2.2.22.50 VSI Mailbox Receive Queue Base Address Low - VSI_MBX_ARQBAL[VSI] (0x00225000 + 0x4*VSI, VSI=0...767; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.60.

13.2.2.22.51 VSI Mailbox Receive Queue Base Address High - VSI_MBX_ARQBAH[VSI] (0x00226000 + 0x4*VSI, VSI=0...767; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.61.

13.2.2.22.52 VSI Mailbox Receive Queue Length - VSI_MBX_ARQLEN[VSI] (0x00227000 + 0x4*VSI, VSI=0...767; RW)

This register specifies the receive queue length. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.62.

13.2.2.22.53 VSI Mailbox Receive Head - VSI_MBX_ARQH[VSI] (0x00228000 + 0x4*VSI, VSI=0...767; RW)

Admin receive queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.63.

13.2.2.22.54 VSI Mailbox Receive Tail - VSI_MBX_ARQT[VSI] (0x00229000 + 0x4*VSI, VSI=0...767; RW)

Admin receive queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.64.

13.2.2.22.55 VF Mailbox Transmit Queue Base Address Low - VF_MBX_ATQBAL[VF] (0x0022A000 + 0x4*VF, VF=0...255; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ATQBAL_LSB	5:0	0x0	RSV	N/A	**Admin Transmit Queue Base Address Low LSB** Tied to zero to achieve alignment.
ATQBAL	31:6	0x0	RW	UNDEFINED	**Admin Transmit Queue Base Address Low** Transmit descriptor base address low. Must be 64-byte aligned (together with ATQBAL_LSB).

13.2.2.22.56 VF Mailbox Transmit Queue Base Address High - VF_MBX_ATQBAH[VF] (0x0022A400 + 0x4*VF, VF=0...255; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ATQBAH	31:0	0x0	RW	UNDEFINED	**Admin Transmit Queue Base Address High** Transmit descriptor base address high.

13.2.2.22.57 VF Mailbox Transmit Queue Length - VF_MBX_ATQLEN[VF] (0x0022A800 + 0x4*VF, VF=0...255; RW)

This register sets the size of the ring. Maximum size is 1024.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ATQLEN	9:0	0x0	RW	UNDEFINED	**Admin Transmit Queue Length** Descriptor ring length. Max size is 1024.
RESERVED	27:10	0x0	RSV	N/A	Reserved.
ATQVFE	28	0b	RW	UNDEFINED	**Admin Transmit Queue VF Error** VF error. This bit is set by firmware on a PF queue when one of its VFs had an Admin Queue error.
ATQOVFL	29	0b	RW	UNDEFINED	**Admin Transmit Queue Overflow** Overflow error. This bit is set by firmware when a message was lost because there was no room on the queue.
ATQCRIT	30	0b	RW	UNDEFINED	**Admin Transmit Queue Critical** Critical error. This bit is set by firmware when a critical error has been detected on this queue.
ATQENABLE	31	0b	RW	UNDEFINED	**Admin Transmit Queue Enable** Enable bit. This bit is set by driver to indicate that the queue is active. When setting the enable bit, software should initialize all other fields. This flag is implemented by a FF and cleared by PFR.

13.2.2.22.58 VF Mailbox Transmit Head - VF_MBX_ATQH[VF] (0x0022AC00 + 0x4*VF, VF=0...255; RW)

Admin transmit queue head pointer.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ATQH	9:0	0x0	RW	UNDEFINED	**Admin Transmit Queue Head** Transmit queue head pointer. At queue initialization, the software clears the head pointer. During nominal operation, the firmware increments the head following command execution.
RESERVED	31:10	0x0	RSV	N/A	Reserved.

13.2.2.22.59 VF Mailbox Transmit Tail - VF_MBX_ATQT[VF] (0x0022B000 + 0x4*VF, VF=0...255; RW)

Admin transmit queue tail pointer.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ATQT	9:0	0x0	RW	UNDEFINED	**Admin Transmit Queue Tail** Transmit queue tail pointer. Incremented to indicate that there are new valid descriptors on the ring. Software can only write to this register once both transmit and receive queues are properly initialized. And clear to zero at queue initialization.
RESERVED	31:10	0x0	RSV	N/A	Reserved.

13.2.2.22.60 VF Mailbox Receive Queue Base Address Low - VF_MBX_ARQBAL[VF] (0x0022B400 + 0x4*VF, VF=0...255; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ARQBAL_LSB	5:0	0x0	RO	N/A	**Admin Receive Queue Base Address Low LSB** Tied to zero to achieve alignment.
ARQBAL	31:6	0x0	RW	UNDEFINED	**Admin Receive Queue Base Address Low** Receive descriptor base address low. Must be 64-byte aligned (together with ARQBAL_LSB).

13.2.2.22.61 VF Mailbox Receive Queue Base Address High - VF_MBX_ARQBAH[VF] (0x0022B800 + 0x4*VF, VF=0...255; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ARQBAH	31:0	0x0	RW	UNDEFINED	**Admin Receive Queue Base Address High** Receive descriptor base address high.

13.2.2.22.62 VF Mailbox Receive Queue Length - VF_MBX_ARQLEN[VF] (0x0022BC00 + 0x4*VF, VF=0...255; RW)

This register specifies the receive queue length. Maximum size is 1024.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ARQLEN	9:0	0x0	RW	UNDEFINED	**Admin Receive Queue Length** Descriptor ring length. Max size is 1024.
RESERVED	27:10	0x0	RSV	N/A	Reserved.
ARQVFE	28	0b	RW	UNDEFINED	**Admin Receive Queue VF Error** VF error. This bit is set by firmware on a PF queue when one of its VFs had an Admin Queue error.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ARQOVFL	29	0b	RW	UNDEFINED	**Admin Receive Queue Overflow** Overflow error. This bit is set by firmware when a message was lost because there was no room on the queue.
ARQCRIT	30	0b	RW	UNDEFINED	**Admin Receive Queue Critical** Critical error. This bit is set by firmware when a critical error has been detected on this queue.
ARQENABLE	31	0b	RW	UNDEFINED	**Admin Receive Queue Enable** Enable bit. This bit is set by driver to indicate that the queue is active. When setting the enable bit, software should initialize all other fields. This flag is implemented by a FF and cleared by PFR.

13.2.2.22.63 VF Mailbox Receive Head - VF_MBX_ARQH[VF] (0x0022C000 + 0x4*VF, VF=0...255; RW)

Admin receive queue head pointer.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ARQH	9:0	0x0	RW	UNDEFINED	**Admin Receive Queue Head** Receive queue head pointer. At queue initialization, the software clears the head pointer. During nominal operation, the firmware increments the head following command execution.
RESERVED	31:10	0x0	RSV	N/A	Reserved.

13.2.2.22.64 VF Mailbox Receive Tail - VF_MBX_ARQT[VF] (0x0022C400 + 0x4*VF, VF=0...255; RW)

Admin receive queue tail pointer.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ARQT	9:0	0x0	RW	UNDEFINED	**Admin Receive Queue Tail** Receive queue tail pointer. Incremented to indicate that there are new valid descriptors on the ring. Software can only write to this register once the queue is fully configured. And clear to zero at queue initialization.
RESERVED	31:10	0x0	RSV	N/A	Reserved.

13.2.2.22.65 VF CPM Mailbox Transmit Queue Base Address Low - VF_MBX_CPM_ATQBAL[VF128] (0x0022C800 + 0x4*VF128, VF128=0...127; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.55.

13.2.2.22.66 VF CPM Mailbox Transmit Queue Base Address High - VF_MBX_CPM_ATQBAH[VF128] (0x0022CA00 + 0x4*VF128, VF128=0...127; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.56.

13.2.2.22.67 VF CPM Mailbox Transmit Queue Length - VF_MBX_CPM_ATQLEN[VF128] (0x0022CC00 + 0x4*VF128, VF128=0...127; RW)

This register sets the size of the ring. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.57.

13.2.2.22.68 VF CPM Mailbox Transmit Head - VF_MBX_CPM_ATQH[VF128] (0x0022CE00 + 0x4*VF128, VF128=0...127; RW)

Admin transmit queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.58.

13.2.2.22.69 VF CPM Mailbox Transmit Tail - VF_MBX_CPM_ATQT[VF128] (0x0022D000 + 0x4*VF128, VF128=0...127; RW)

Admin transmit queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.59.

13.2.2.22.70 VF CPM Mailbox Receive Queue Base Address Low - VF_MBX_CPM_ARQBAL[VF128] (0x0022D200 + 0x4*VF128, VF128=0...127; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.60.

13.2.2.22.71 VF CPM Mailbox Receive Queue Base Address High - VF_MBX_CPM_ARQBAH[VF128] (0x0022D400 + 0x4*VF128, VF128=0...127; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.61.

13.2.2.22.72 VF CPM Mailbox Receive Queue Length - VF_MBX_CPM_ARQLEN[VF128] (0x0022D600 + 0x4*VF128, VF128=0...127; RW)

This register specifies the receive queue length. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.62.

13.2.2.22.73 VF CPM Mailbox Receive Head - VF_MBX_CPM_ARQH[VF128] (0x0022D800 + 0x4*VF128, VF128=0...127; RW)

Admin receive queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.63.

13.2.2.22.74 VF CPM Mailbox Receive Tail - VF_MBX_CPM_ARQT[VF128] (0x0022DA00 + 0x4*VF128, VF128=0...127; RW)

Admin receive queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.64.

13.2.2.22.75 VF HLP Mailbox Transmit Queue Base Address Low - VF_MBX_HLP_ATQBAL[VF16] (0x0022DC00 + 0x4*VF16, VF16=0...15; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.55.

13.2.2.22.76 VF HLP Mailbox Transmit Queue Base Address High - VF_MBX_HLP_ATQBAH[VF16] (0x0022DC40 + 0x4*VF16, VF16=0...15; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.56.

13.2.2.22.77 VF HLP Mailbox Transmit Queue Length - VF_MBX_HLP_ATQLEN[VF16] (0x0022DC80 + 0x4*VF16, VF16=0...15; RW)

This register sets the size of the ring. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.57.

13.2.2.22.78 VF HLP Mailbox Transmit Head - VF_MBX_HLP_ATQH[VF16] (0x0022DCC0 + 0x4*VF16, VF16=0...15; RW)

Admin transmit queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.58.

13.2.2.22.79 VF HLP Mailbox Transmit Tail - VF_MBX_HLP_ATQT[VF16] (0x0022DD00 + 0x4*VF16, VF16=0...15; RW)

Admin transmit queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.59.

13.2.2.22.80 VF HLP Mailbox Receive Queue Base Address Low - VF_MBX_HLP_ARQBAL[VF16] (0x0022DD40 + 0x4*VF16, VF16=0...15; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.60.

13.2.2.22.81 VF HLP Mailbox Receive Queue Base Address High - VF_MBX_HLP_ARQBAH[VF16] (0x0022DD80 + 0x4*VF16, VF16=0...15; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.61.

13.2.2.22.82 VF HLP Mailbox Receive Queue Length - VF_MBX_HLP_ARQLEN[VF16] (0x0022DDC0 + 0x4*VF16, VF16=0...15; RW)

This register specifies the receive queue length. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.62.

13.2.2.22.83 VF HLP Mailbox Receive Head - VF_MBX_HLP_ARQH[VF16] (0x0022DE00 + 0x4*VF16, VF16=0...15; RW)

Admin receive queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.63.

13.2.2.22.84 VF HLP Mailbox Receive Tail - VF_MBX_HLP_ARQT[VF16] (0x0022DE40 + 0x4*VF16, VF16=0...15; RW)

Admin receive queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.64.

13.2.2.22.85 VF PSM Mailbox Transmit Queue Base Address Low - VF_MBX_PSM_ATQBAL[VF16] (0x0022DE80 + 0x4*VF16, VF16=0...15; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.55.

13.2.2.22.86 VF PSM Mailbox Transmit Queue Base Address High - VF_MBX_PSM_ATQBAH[VF16] (0x0022DEC0 + 0x4*VF16, VF16=0...15; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.56.

13.2.2.22.87 VF PSM Mailbox Transmit Queue Length - VF_MBX_PSM_ATQLEN[VF16] (0x0022DF00 + 0x4*VF16, VF16=0...15; RW)

This register sets the size of the ring. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.57.

13.2.2.22.88 VF PSM Mailbox Transmit Head - VF_MBX_PSM_ATQH[VF16] (0x0022DF40 + 0x4*VF16, VF16=0...15; RW)

Admin transmit queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.58.

13.2.2.22.89 VF PSM Mailbox Transmit Tail - VF_MBX_PSM_ATQT[VF16] (0x0022DF80 + 0x4*VF16, VF16=0...15; RW)

Admin transmit queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.59.

13.2.2.22.90 VF PSM Mailbox Receive Queue Base Address Low - VF_MBX_PSM_ARQBAL[VF16] (0x0022DFC0 + 0x4*VF16, VF16=0...15; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.60.

13.2.2.22.91 VF PSM Mailbox Receive Queue Base Address High - VF_MBX_PSM_ARQBAH[VF16] (0x0022E000 + 0x4*VF16, VF16=0...15; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.61.

13.2.2.22.92 VF PSM Mailbox Receive Queue Length - VF_MBX_PSM_ARQLEN[VF16] (0x0022E040 + 0x4*VF16, VF16=0...15; RW)

This register specifies the receive queue length. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.62.

13.2.2.22.93 VF PSM Mailbox Receive Head - VF_MBX_PSM_ARQH[VF16] (0x0022E080 + 0x4*VF16, VF16=0...15; RW)

Admin receive queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.63.

13.2.2.22.94 VF PSM Mailbox Receive Tail - VF_MBX_PSM_ARQT[VF16] (0x0022E0C0 + 0x4*VF16, VF16=0...15; RW)

Admin receive queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.64.

13.2.2.22.95 PF Mailbox Transmit Queue Base Address Low - PF_MBX_ATQBAL (0x0022E100; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.55.

13.2.2.22.96 PF Mailbox Transmit Queue Base Address High - PF_MBX_ATQBAH (0x0022E180; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.56.

13.2.2.22.97 PF Mailbox Transmit Queue Length - PF_MBX_ATQLEN (0x0022E200; RW)

This register sets the size of the ring. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.57.

13.2.2.22.98 PF Mailbox Transmit Head - PF_MBX_ATQH (0x0022E280; RW)

Admin transmit queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.58.

13.2.2.22.99 PF Mailbox Transmit Tail - PF_MBX_ATQT (0x0022E300; RW)

Admin transmit queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.59.

13.2.2.22.100 PF Mailbox Receive Queue Base Address Low - PF_MBX_ARQBAL (0x0022E380; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.60.

13.2.2.22.101 PF Mailbox Receive Queue Base Address High - PF_MBX_ARQBAH (0x0022E400; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.61.

13.2.2.22.102 PF Mailbox Receive Queue Length - PF_MBX_ARQLEN (0x0022E480; RW)

This register specifies the receive queue length. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.62.

13.2.2.22.103 PF Mailbox Receive Head - PF_MBX_ARQH (0x0022E500; RW)

Admin receive queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.63.

13.2.2.22.104 PF Mailbox Receive Tail - PF_MBX_ARQT (0x0022E580; RW)

Admin receive queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.64.

13.2.2.22.105 PF0 CPM Mailbox Transmit Queue Base Address Low - PF0_MBX_CPM_ATQBAL (0x0022E5C0; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.55.

13.2.2.22.106 PF0 CPM Mailbox Transmit Queue Base Address High - PF0_MBX_CPM_ATQBAH (0x0022E5C4; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.56.

13.2.2.22.107 PF0 CPM Mailbox Transmit Queue Length - PF0_MBX_CPM_ATQLEN (0x0022E5C8; RW)

This register sets the size of the ring. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.57.

13.2.2.22.108 PF0 CPM Mailbox Transmit Head - PF0_MBX_CPM_ATQH (0x0022E5CC; RW)

Admin transmit queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.58.

13.2.2.22.109 PF0 CPM Mailbox Transmit Tail - PF0_MBX_CPM_ATQT (0x0022E5D0; RW)

Admin transmit queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.59.

13.2.2.22.110 PF0 CPM Mailbox Receive Queue Base Address Low - PF0_MBX_CPM_ARQBAL (0x0022E5D4; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.60.

13.2.2.22.111 PF0 CPM Mailbox Receive Queue Base Address High - PF0_MBX_CPM_ARQBAH (0x0022E5D8; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.61.

13.2.2.22.112 PF0 CPM Mailbox Receive Queue Length - PF0_MBX_CPM_ARQLEN (0x0022E5DC; RW)

This register specifies the receive queue length. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.62.

13.2.2.22.113 PF0 CPM Mailbox Receive Head - PF0_MBX_CPM_ARQH (0x0022E5E0; RW)

Admin receive queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.63.

13.2.2.22.114 PF0 CPM Mailbox Receive Tail - PF0_MBX_CPM_ARQT (0x0022E5E4; RW)

Admin receive queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.64.

13.2.2.22.115 PF0 HLP Mailbox Transmit Queue Base Address Low - PF0_MBX_HLP_ATQBAL (0x0022E5E8; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.55.

13.2.2.22.116 PF0 HLP Mailbox Transmit Queue Base Address High - PF0_MBX_HLP_ATQBAH (0x0022E5EC; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.56.

13.2.2.22.117 PF0 HLP Mailbox Transmit Queue Length - PF0_MBX_HLP_ATQLEN (0x0022E5F0; RW)

This register sets the size of the ring. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.57.

13.2.2.22.118 PF0 HLP Mailbox Transmit Head - PF0_MBX_HLP_ATQH (0x0022E5F4; RW)

Admin transmit queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.58.

13.2.2.22.119 PF0 HLP Mailbox Transmit Tail - PF0_MBX_HLP_ATQT (0x0022E5F8; RW)

Admin transmit queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.59.

13.2.2.22.120 PF0 HLP Mailbox Receive Queue Base Address Low - PF0_MBX_HLP_ARQBAL (0x0022E5FC; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.60.

13.2.2.22.121 PF0 HLP Mailbox Receive Queue Base Address High - PF0_MBX_HLP_ARQBAH (0x0022E600; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.61.

13.2.2.22.122 PF0 HLP Mailbox Receive Queue Length - PF0_MBX_HLP_ARQLEN (0x0022E604; RW)

This register specifies the receive queue length. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.62.

13.2.2.22.123 PF0 HLP Mailbox Receive Head - PF0_MBX_HLP_ARQH (0x0022E608; RW)

Admin receive queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.63.

13.2.2.22.124 PF0 HLP Mailbox Receive Tail - PF0_MBX_HLP_ARQT (0x0022E60C; RW)

Admin receive queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.64.

13.2.2.22.125 PF0 PSM Mailbox Transmit Queue Base Address Low - PF0_MBX_PSM_ATQBAL (0x0022E610; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.55.

13.2.2.22.126 PF0 PSM Mailbox Transmit Queue Base Address High - PF0_MBX_PSM_ATQBAH (0x0022E614; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.56.

13.2.2.22.127 PF0 PSM Mailbox Transmit Queue Length - PF0_MBX_PSM_ATQLEN (0x0022E618; RW)

This register sets the size of the ring. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.57.

13.2.2.22.128 PF0 PSM Mailbox Transmit Head - PF0_MBX_PSM_ATQH (0x0022E61C; RW)

Admin transmit queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.58.

13.2.2.22.129 PF0 PSM Mailbox Transmit Tail - PF0_MBX_PSM_ATQT (0x0022E620; RW)

Admin transmit queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.59.

13.2.2.22.130 PF0 PSM Mailbox Receive Queue Base Address Low - PF0_MBX_PSM_ARQBAL (0x0022E624; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.60.

13.2.2.22.131 PF0 PSM Mailbox Receive Queue Base Address High - PF0_MBX_PSM_ARQBAH (0x0022E628; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.61.

13.2.2.22.132 PF0 PSM Mailbox Receive Queue Length - PF0_MBX_PSM_ARQLEN (0x0022E62C; RW)

This register specifies the receive queue length. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.62.

13.2.2.22.133 PF0 PSM Mailbox Receive Head - PF0_MBX_PSM_ARQH (0x0022E630; RW)

Admin receive queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.63.

13.2.2.22.134 PF0 PSM Mailbox Receive Tail - PF0_MBX_PSM_ARQT (0x0022E634; RW)

Admin receive queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.64.

13.2.2.22.135 PF0 CPM Sideband Transmit Queue Base Address Low - PF0_SB_CPM_ATQBAL (0x0022E638; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.55.

13.2.2.22.136 PF0 CPM Sideband Transmit Queue Base Address High - PF0_SB_CPM_ATQBAH (0x0022E63C; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.56.

13.2.2.22.137 PF0 CPM Sideband Transmit Queue Length - PF0_SB_CPM_ATQLEN (0x0022E640; RW)

This register sets the size of the ring. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.57.

13.2.2.22.138 PF0 CPM Sideband Transmit Head - PF0_SB_CPM_ATQH (0x0022E644; RW)

Admin transmit queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.58.

13.2.2.22.139 PF0 CPM Sideband Transmit Tail - PF0_SB_CPM_ATQT (0x0022E648; RW)

Admin transmit queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.59.

13.2.2.22.140 PF0 CPM Sideband Receive Queue Base Address Low - PF0_SB_CPM_ARQBAL (0x0022E64C; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.60.

13.2.2.22.141 PF0 CPM Sideband Receive Queue Base Address High - PF0_SB_CPM_ARQBAH (0x0022E650; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.61.

13.2.2.22.142 PF0 CPM Sideband Receive Queue Length - PF0_SB_CPM_ARQLEN (0x0022E654; RW)

This register specifies the receive queue length. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.62.

13.2.2.22.143 PF0 CPM Sideband Receive Head - PF0_SB_CPM_ARQH (0x0022E658; RW)

Admin receive queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.63.

13.2.2.22.144 PF0 CPM Sideband Receive Tail - PF0_SB_CPM_ARQT (0x0022E65C; RW)

Admin receive queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.64.

13.2.2.22.145 VF CPM Sideband Transmit Queue Base Address Low - VF_SB_CPM_ATQBAL[VF128] (0x0022E800 + 0x4*VF128, VF128=0...127; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.55.

13.2.2.22.146 VF CPM Sideband Transmit Queue Base Address High - VF_SB_CPM_ATQBAH[VF128] (0x0022EA00 + 0x4*VF128, VF128=0...127; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.56.

13.2.2.22.147 VF CPM Sideband Transmit Queue Length - VF_SB_CPM_ATQLEN[VF128] (0x0022EC00 + 0x4*VF128, VF128=0...127; RW)

This register sets the size of the ring. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.57.

13.2.2.22.148 VF CPM Sideband Transmit Head - VF_SB_CPM_ATQH[VF128] (0x0022EE00 + 0x4*VF128, VF128=0...127; RW)

Admin transmit queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.58.

13.2.2.22.149 VF CPM Sideband Transmit Tail - VF_SB_CPM_ATQT[VF128] (0x0022F000 + 0x4*VF128, VF128=0...127; RW)

Admin transmit queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.59.

13.2.2.22.150 VF CPM Sideband Receive Queue Base Address Low - VF_SB_CPM_ARQBAL[VF128] (0x0022F200 + 0x4*VF128, VF128=0...127; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.60.

13.2.2.22.151 VF CPM Sideband Receive Queue Base Address High - VF_SB_CPM_ARQBAH[VF128] (0x0022F400 + 0x4*VF128, VF128=0...127; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.61.

13.2.2.22.152 VF CPM Sideband Receive Queue Length - VF_SB_CPM_ARQLEN[VF128] (0x0022F600 + 0x4*VF128, VF128=0...127; RW)

This register specifies the receive queue length. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.62.

13.2.2.22.153 VF CPM Sideband Receive Head - VF_SB_CPM_ARQH[VF128] (0x0022F800 + 0x4*VF128, VF128=0...127; RW)

Admin receive queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.63.

13.2.2.22.154 VF CPM Sideband Receive Tail - VF_SB_CPM_ARQT[VF128] (0x0022FA00 + 0x4*VF128, VF128=0...127; RW)

Admin receive queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.64.

13.2.2.22.155 PF Sideband Transmit Queue Base Address Low - PF_SB_ATQBAL (0x0022FC00; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.55.

13.2.2.22.156 PF Sideband Transmit Queue Base Address High - PF_SB_ATQBAH (0x0022FC80; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.56.

13.2.2.22.157 PF Sideband Transmit Queue Length - PF_SB_ATQLEN (0x0022FD00; RW)

This register sets the size of the ring. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.57.

13.2.2.22.158 PF Sideband Transmit Head - PF_SB_ATQH (0x0022FD80; RW)

Admin transmit queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.58.

13.2.2.22.159 PF Sideband Transmit Tail - PF_SB_ATQT (0x0022FE00; RW)

Admin transmit queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.59.

13.2.2.22.160 PF Sideband Receive Queue Base Address Low - PF_SB_ARQBAL (0x0022FE80; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.60.

13.2.2.22.161 PF Sideband Receive Queue Base Address High - PF_SB_ARQBAH (0x0022FF00; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.61.

13.2.2.22.162 PF Sideband Receive Queue Length - PF_SB_ARQLEN (0x0022FF80; RW)

This register specifies the receive queue length. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.62.

13.2.2.22.163 PF Sideband Receive Head - PF_SB_ARQH (0x00230000; RW)

Admin receive queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.63.

13.2.2.22.164 PF Sideband Receive Tail - PF_SB_ARQT (0x00230080; RW)

Admin receive queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.64.

13.2.2.22.165 PF0 HLP Sideband Transmit Queue Base Address Low - PF0_SB_HLP_ATQBAL (0x002300C0; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.55.

13.2.2.22.166 PF0 HLP Sideband Transmit Queue Base Address High - PF0_SB_HLP_ATQBAH (0x002300C4; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.56.

13.2.2.22.167 PF0 HLP Sideband Transmit Queue Length - PF0_SB_HLP_ATQLEN (0x002300C8; RW)

This register sets the size of the ring. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.57.

13.2.2.22.168 PF0 HLP Sideband Transmit Head - PF0_SB_HLP_ATQH (0x002300CC; RW)

Admin transmit queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.58.

13.2.2.22.169 PF0 HLP Sideband Transmit Tail - PF0_SB_HLP_ATQT (0x002300D0; RW)

Admin transmit queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.59.

13.2.2.22.170 PF0 HLP Sideband Receive Queue Base Address Low - PF0_SB_HLP_ARQBAL (0x002300D4; RW)

This register contains the lower bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.60.

13.2.2.22.171 PF0 HLP Sideband Receive Queue Base Address High - PF0_SB_HLP_ARQBAH (0x002300D8; RW)

This register contains the higher bits of the 64-bit descriptor base address.

Field definitions are the same as those defined in Section 13.2.2.22.61.

13.2.2.22.172 PF0 HLP Sideband Receive Queue Length - PF0_SB_HLP_ARQLEN (0x002300DC; RW)

This register specifies the receive queue length. Maximum size is 1024.

Field definitions are the same as those defined in Section 13.2.2.22.62.

13.2.2.22.173 PF0 HLP Sideband Receive Head - PF0_SB_HLP_ARQH (0x002300E0; RW)

Admin receive queue head pointer.

Field definitions are the same as those defined in Section 13.2.2.22.63.

13.2.2.22.174 PF0 HLP Sideband Receive Tail - PF0_SB_HLP_ARQT (0x002300E4; RW)

Admin receive queue tail pointer.

Field definitions are the same as those defined in Section 13.2.2.22.64.

13.2.2.22.175 PF SB HLP Remote Device Control Register - PF0_SB_HLP_REM_DEV_CTL (0x002300E8; RW)

Bit mapping of the remote devices allowed to be accessed by HLP PF driver.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DEST_EN	15:0	0x0	RW	UNDEFINED	**Destination Enable** Bit mapping of the sideband remote devices allowed to be accessed by the software driver.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.22.176 VF SB CPM Remote Device Control Register - VF_SB_CPM_REM_DEV_CTL (0x002300EC; RW)

Bit mapping of the remote devices allowed to be accessed by CPM VF driver.

Field definitions are the same as those defined in Section 13.2.2.22.175.

13.2.2.22.177 PF SB Remote Device Control Register - PF_SB_REM_DEV_CTL (0x002300F0; RW)

Bit mapping of the remote devices allowed to be accessed by CPK PF driver.

Field definitions are the same as those defined in Section 13.2.2.22.175.

13.2.2.22.178 PF SB CPM Remote Device Control Register - PF0_SB_CPM_REM_DEV_CTL (0x002300F4; RW)

Bit mapping of the remote devices allowed to be accessed by CPM PF driver.

Field definitions are the same as those defined in Section 13.2.2.22.175.

13.2.2.22.179 SB Remote Device Destination Register - SB_REM_DEV_DEST[n] (0x002300F8 + 0x4*n, n=0...7; RW)

This register contains the destination software driver that the SB remote device is mapped to (per remote device source)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DEST	3:0	0x0	RW	UNDEFINED	**Destination** Destination software driver of this sideband remote device. 0x0 = CPK driver 0x1 = CPM driver 0x2 = HLP driver All other values are reserved.
RESERVED	30:4	0x0	RSV	N/A	Reserved.
DEST_VALID	31	0b	RW	UNDEFINED	**Destination Valid**

13.2.2.22.180 PF VF Control Register - VP_MBX_PF_VF_CTRL[VSI] (0x00230800 + 0x4*VSI, VSI=0...767; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
QUEUE_EN	0	1b	RW	UNDEFINED	**Queue Enable** A PF can disable a VF's transmit and receive Admin Queues by clearing this bit.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.22.181 PF VF Control Register - VP_MBX_CPM_PF_VF_CTRL[VP128] (0x00231800 + 0x4*VP128, VP128=0...127; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
QUEUE_EN	0	1b	RW	UNDEFINED	**Queue Enable** A PF can disable a VF's transmit and receive Admin Queues by clearing this bit.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.22.182 PF VF Control Register - VP_MBX_HLP_PF_VF_CTRL[VP16] (0x00231A00 + 0x4*VP16, VP16=0...15; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
QUEUE_EN	0	1b	RW	UNDEFINED	**Queue Enable** A PF can disable a VF's transmit and receive Admin Queues by clearing this bit.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.22.183 PF VF Control Register - VP_MBX_PSM_PF_VF_CTRL[VP16] (0x00231A40 + 0x4*VP16, VP16=0...15; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
QUEUE_EN	0	1b	RW	UNDEFINED	**Queue Enable** A PF can disable a VF's transmit and receive Admin Queues by clearing this bit.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.22.184 PF VF Control Register - VP_SB_CPM_PF_VF_CTRL[VP128] (0x00231C00 + 0x4*VP128, VP128=0...127; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
QUEUE_EN	0	1b	RW	UNDEFINED	**Queue Enable** A PF can disable a VF's transmit and receive Admin Queues by clearing this bit.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.22.185 PF VF Control Register - GL_MBX_PASID (0x00231EC0; RW)

This register contains MBX PASID related configurations.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PASID_MODE	0	0b	RW	UNDEFINED	**PASID Mode** When set, the E810 works in PASID mode, and MBX should handle its queues accordingly.
PASID_MODE_VALID	1	0b	RW	N/A	**PASID Mode Valid** Validates the PASID_MODE bit (for software use only).
RESERVED	31:2	0x0	RSV	UNDEFINED	Reserved.

13.2.2.23 PF - Statistics Registers

Statistics counters. Refer to Section 9.6 for information regarding wide counters, clearing counters, sampling points, and counter consistency rules.

Note: Wide counters (48 bits) are represented as two registers (high and low) containing the 16 most significant bits and 32 least significant bits, respectively. They are implemented as a 64-bit register. Atomicity is only guaranteed when reading both parts in one 64-bit read.

Note: Per-function registers are implemented as an array of 144 registers. Elements 0-127 correspond to VFs 0-127 and elements 128-143 are used for PFs 0-15.

13.2.2.23.1 PORT TC Transmit Byte Count - TPB_PRTTPB_STAT_TC_BYTES_SENT[n] (0x00099094 + 0x4*n, n=0...63; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCCNT	31:0	0x0	RW1C	DYNAMIC	**TC Count** Port per-TC byte count. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.2 PORT Transmit Packet Count - TPB_PRTTPB_STAT_PKT_SENT[n] (0x00099470 + 0x4*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PKTCNT	31:0	0x0	RW1C	DYNAMIC	**Packet Count** Port packet counter.

13.2.2.23.3 Port (Line) Receive Drop Counter - PRTRPB_RDPC (0x000AC260; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CRCERRS	31:0	0x0	RW1C	DYNAMIC	**CRC Errors** Port drop counter. Counts all packets that were dropped due to shortage of storage space.

13.2.2.23.4 Port (LB) Receive Drop Counter - PRTRPB_LDPC (0x000AC280; RWC)

Field definitions are the same as those defined in Section 13.2.2.23.3.

13.2.2.23.5 VSI Received Discard Packet Count - GLV_RDPC[n] (0x00294C04 + 0x4*n, n=0...767; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDPC	31:0	0x0	RWC	UNDEFINED	**Received Discard Packet Count** Counts (per VSI) packets that were drop due to no descriptors in host queue or other drops in the pipe. Pipe drops includes Flow Director, ACL, and packets directed to invalid receive queues drops. For EMP VSIs, it counts dropped packets due to no EMP buffer space.

13.2.2.23.6 Per VSI Error Drops - GLV_REPC[n] (0x00295804 + 0x4*n, n=0...767; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
NO_DESC_CNT	15:0	0x0	RW	UNDEFINED	**No Descriptors Count** Counts packets dropped from this VSI due to no descriptors available. For VSIs associated with the EMP, counts drops due to EMP buffer full. Stuck at 0xFFFF.
ERROR_CNT	31:16	0x0	RW	UNDEFINED	**Error Count** Counts packet dropped from this VSI due to the following cases: • Packet size is larger than RXMAX of the queue. • Receive descriptor Unsupported Request on the PCI or internal Dummy completion. • Packets directed to disabled receive queues. • Packets dropped due to VM reset, VF reset, or PF reset. Stuck at 0xFFFF

Note: Writing to this register clears both the counters. There is no way to clear a single counter.

13.2.2.23.7 VSI Good Octets Transmit Count Low - GLV_GOTCL[n] (0x00300000 + 0x8*n, n=0...767; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GOTCL	31:0	0x0	RWC	UNDEFINED	**Good Octet Transmit Count Low** Transmit octet count. Counts the number of bytes transmitted by this VSI. Lower 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.8 VSI Good Octets Transmit Count High - GLV_GOTCH[n] (0x00300004 + 0x8*n, n=0...767; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GOTCH	7:0	0x0	RWC	UNDEFINED	**Good Octet Transmit Count High** Transmit octet count. Counts the number of bytes transmitted by this VSI. Higher 8 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.9 Switch Good Octets Transmit Count Low - GLSW_GOTCL[n] (0x00302000 + 0x8*n, n=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GOTCL	31:0	0x0	RWC	UNDEFINED	**Good Octet Transmit Count Low** Transmit octet count. Counts the number of bytes transmitted. Lower 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.10 Switch Good Octets Transmit Count High - GLSW_GOTCH[n] (0x00302004 + 0x8*n, n=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GOTCH	7:0	0x0	RWC	UNDEFINED	**Good Octet Transmit Count High** Transmit octet count. Counts the number of bytes transmitted. Higher 8 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.11 VEB VLAN Transmit Byte Count Low - GL_STAT_SWR_GOTCL[n] (0x00304000 + 0x8*n, n=0...127; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VLBCL	31:0	0x0	RWC	UNDEFINED	**VLAN Byte Count Low** VEB per VLAN byte count low. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.12 VEB VLAN Transmit Byte Count High - GL_STAT_SWR_GOTCH[n] (0x00304004 + 0x8*n, n=0...127; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VLBCH	7:0	0x0	RWC	UNDEFINED	**VLAN Byte Count High** VEB per VLAN byte count High. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.13 VEB UP Transmit Byte Count Low - GLVEBUP_TBCL[n,m] (0x00306000 + 0x8*n + 0x40*m, n=0...7, m=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UPBCL	31:0	0x0	RWC	UNDEFINED	**UP Byte Count Low** UP transmit byte count low. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.14 VEB UP Transmit Byte Count High - GLVEBUP_TBCH[n,m] (0x00306004 + 0x8*n + 0x40*m, n=0...7, m=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UPBCH	7:0	0x0	RWC	UNDEFINED	**UP Byte Count High** UP transmit byte count High. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.15 VEB UP Transmit Packet Count Low - GLVEBUP_TPCL[n,m] (0x00308000 + 0x8*n + 0x40*m, n=0...7, m=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UPPCL	31:0	0x0	RWC	UNDEFINED	**UP Packet Count Low** VEB per UP transmit packets count low. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.16 VEB UP Transmit Packet Count High - GLVEBUP_TPCH[n,m] (0x00308004 + 0x8*n + 0x40*m, n=0...7, m=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UPPCH	7:0	0x0	RWC	UNDEFINED	**UP Packet Count High** VEB per UP transmit packets count high. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.17 VSI Unicast Packets Transmit Count Low - GLV_UPTCL[n] (0x0030A000 + 0x8*n, n=0...767; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UPTCL	31:0	0x0	RWC	UNDEFINED	**Unicast Packets Transmit Count Low** Transmit unicast packet count. Counts the number of unicast packets transmitted by this VSI. Lower 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.18 VSI Unicast Packets Transmit Count High - GLV_UPTCH[n] (0x0030A004 + 0x8*n, n=0...767; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GLVUPTCH	7:0	0x0	RWC	UNDEFINED	**Unicast Packets Transmit Count High** Transmit unicast packet count. Counts the number of unicast packets transmitted by this VSI. Higher 8 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.19 VSI Multicast Packets Transmit Count Low - GLV_MPTCL[n] (0x0030C000 + 0x8*n, n=0...767; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MPTCL	31:0	0x0	RWC	UNDEFINED	**Multicast Packets Transmit Count Low** Transmit multicast packet count. Counts the number of multicast packets transmitted by this VSI. Lower 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.20 VSI Multicast Packets Transmit Count High - GLV_MPTCH[n] (0x0030C004 + 0x8*n, n=0...767; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MPTCH	7:0	0x0	RWC	UNDEFINED	**Multicast Packets Transmit Count High** Transmit multicast packet count. Counts the number of multicast packets transmitted by this VSI. Higher 8 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.21 VSI Broadcast Packets Transmit Count Low - GLV_BPTCL[n] (0x0030E000 + 0x8*n, n=0...767; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BPTCL	31:0	0x0	RWC	UNDEFINED	**Broadcast Packets Transmit Count Low** Transmit broadcast packet count. Counts the number of broadcast packets transmitted by this VSI. Lower 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.22 VSI Broadcast Packets Transmit Count High - GLV_BPTCH[n] (0x0030E004 + 0x8*n, n=0...767; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BPTCH	7:0	0x0	RWC	UNDEFINED	**Broadcast Packets Transmit Count High** Transmit broadcast packet count. Counts the number of broadcast packets transmitted by this VSI. Higher 8 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.23 Switch Unicast Packets Transmit Count Low - GLSW_UPTCL[n] (0x00310000 + 0x8*n, n=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UPTCL	31:0	0x0	RWC	UNDEFINED	**Unicast Packets Transmit Count Low** Transmit unicast packet count. Counts the number of unicast packets transmitted. Lower 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.24 Switch Unicast Packets Transmit Count High - GLSW_UPTCH[n] (0x00310004 + 0x8*n, n=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UPTCH	7:0	0x0	RWC	UNDEFINED	**Unicast Packets Transmit Count High** Transmit unicast packet count. Counts the number of unicast packets transmitted. Higher 8 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.25 Switch Multicast Packets Transmit Count Low - GLSW_MPTCL[n] (0x00310100 + 0x8*n, n=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MPTCL	31:0	0x0	RWC	UNDEFINED	**Multicast Packets Transmit Count Low** Transmit multicast packet count. Counts the number of multicast packets transmitted. Lower 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.26 Switch Multicast Packets Transmit Count High - GLSW_MPTCH[n] (0x00310104 + 0x8*n, n=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MPTCH	7:0	0x0	RWC	UNDEFINED	**Multicast Packets Transmit Count High** Transmit multicast packet count. Counts the number of multicast packets transmitted. Higher 8 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.27 Switch Broadcast Packets Transmit Count Low - GLSW_BPTCL[n] (0x00310200 + 0x8*n, n=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BPTCL	31:0	0x0	RWC	UNDEFINED	**Broadcast Packets Transmit Count Low** Transmit broadcast packet count. Counts the number of broadcast packets transmitted. Lower 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.28 Switch Broadcast Packets Transmit Count High - GLSW_BPTCH[n] (0x00310204 + 0x8*n, n=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BPTCH	7:0	0x0	RWC	UNDEFINED	**Broadcast Packets Transmit Count High** Transmit broadcast packet count. Counts the number of broadcast packets transmitted. Higher 8 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.29 VSI Transmit Error Packet Count - GLV_TEPC[VSI] (0x00312000 + 0x4*VSI, VSI=0...767; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TEPC	31:0	0x0	RWC	UNDEFINED	**Transmit Error Packet Count** Includes: • Packets dropped due to drop resolution of the switch (for example, violation of anti-spoof rules) - i.e. *LAN_EN* = 0 and *LB_EN* = 0. • Packets dropped due to malicious behavior (including In-line IPsec malicious behavior and DSCP enforcement). • TTL <= 0

13.2.2.23.30 Port Storm Control Discarded Count - GLPRT_STDC[n] (0x00340000 + 0x4*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
STDC	31:0	0x0	RWC	UNDEFINED	**Storm Control Discarded Count** Packets dropped due to storm control.

13.2.2.23.31 Switch Good Octets Received Count Low - GLSW_GORCL[n] (0x00341000 + 0x8*n, n=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GORCL	31:0	0x0	RWC	UNDEFINED	**Good Octet Receive Count Low** Receive octet count. Counts the number of bytes received. Lower 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.32 Switch Good Octets Received Count High - GLSW_GORCH[n] (0x00341004 + 0x8*n, n=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GORCH	7:0	0x0	RWC	UNDEFINED	**Good Octet Receive Count High** Receive octet count. Counts the number of bytes received. Higher 8 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.33 VEB VLAN Receive Byte Count Low - GL_STAT_SWR_GORCL[n] (0x00342000 + 0x8*n, n=0...127; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VLBCL	31:0	0x0	RWC	UNDEFINED	**VLAN Byte Count Low** VEB per VLAN Byte count low. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.34 VEB VLAN Receive Byte Count High - GL_STAT_SWR_GORCH[n] (0x00342004 + 0x8*n, n=0...127; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VLBCH	7:0	0x0	RWC	UNDEFINED	**VLAN Byte Count High** VEB per VLAN Byte count high. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.35 VEB UP Receive Byte Count Low - GLVEBUP_RBCL[n,m] (0x00343000 + 0x8*n + 0x40*m, n=0...7, m=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UPBCL	31:0	0x0	RWC	UNDEFINED	**UP Byte Count Low** VEB per UP receive byte count low. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.36 VEB UP Receive Byte Count High - GLVEBUP_RBCH[n,m] (0x00343004 + 0x8*n + 0x40*m, n=0...7, m=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UPBCH	7:0	0x0	RWC	UNDEFINED	**UP Byte Count High** VEB per UP transmit byte count high. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.37 VEB UP Receive Packet Count Low - GLVEBUP_RPCL[n,m] (0x00344000 + 0x8*n + 0x40*m, n=0...7, m=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UPPCL	31:0	0x0	RWC	UNDEFINED	**UP Packet Count Low** VEB per UP receive packets count low. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.38 VEB UP Receive Packet Count High - GLVEBUP_RPCH[n,m] (0x00344004 + 0x8*n + 0x40*m, n=0...7, m=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UPPCH	7:0	0x0	RWC	UNDEFINED	**UP Packet Count High** VEB per UP receive packets count high. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.39 Switch Unicast Packets Received Count Low - GLSW_UPRCL[n] (0x00346000 + 0x8*n, n=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UPRCL	31:0	0x0	RWC	UNDEFINED	**Unicast Packets Received Count Low** Receive unicast packet count. Counts the number of unicast packets received. Lower 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.40 Switch Unicast Packets Received Count High - GLSW_UPRCH[n] (0x00346004 + 0x8*n, n=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UPRCH	7:0	0x0	RWC	UNDEFINED	**Unicast Packets Received Count High** Receive unicast packet count. Counts the number of unicast packets received. Higher 8 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.41 Switch Multicast Packets Received Count Low - GLSW_MPRCL[n] (0x00346100 + 0x8*n, n=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MPRCL	31:0	0x0	RWC	UNDEFINED	**Multicast Packets Received Count Low** Receive multicast packet count. Counts the number of multicast packets received. Lower 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.42 Switch Multicast Packets Received Count High - GLSW_MPRCH[n] (0x00346104 + 0x8*n, n=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MPRCH	7:0	0x0	RWC	UNDEFINED	**Multicast Packets Received Count High** Receive multicast packet count. Counts the number of multicast packets received. Higher 8 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.43 Switch Broadcast Packets Received Count Low - GLSW_BPRCL[n] (0x00346200 + 0x8*n, n=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BPRCL	31:0	0x0	RWC	UNDEFINED	**Broadcast Packets Received Count Low** Receive broadcast packet count. Counts the number of broadcast packets received. Lower 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.44 Switch Broadcast Packets Received Count High - GLSW_BPRCH[n] (0x00346204 + 0x8*n, n=0...31; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BPRCH	7:0	0x0	RWC	UNDEFINED	**Broadcast Packets Received Count High** Receive broadcast packet count. Counts the number of broadcast packets received. Higher 8 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.45 VEB VLAN Unicast Packet Count Low - GL_STAT_SWR_UPCL[n] (0x00347000 + 0x8*n, n=0...127; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VLUPCL	31:0	0x0	RWC	UNDEFINED	**VLAN Unicast Packet Count Low** VEB per VLAN unicast packet count low. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.46 VEB VLAN Unicast Packet Count High - GL_STAT_SWR_UPCH[n] (0x00347004 + 0x8*n, n=0...127; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VLUPCH	7:0	0x0	RWC	UNDEFINED	**VLAN Unicast Packet Count High** VEB per VLAN unicast packet count high. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.47 VEB VLAN Multicast Packet Count Low - GL_STAT_SWR_MPCL[n] (0x00347400 + 0x8*n, n=0...127; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VLMPCL	31:0	0x0	RWC	UNDEFINED	**VLAN Multicast Packet Count Low** VEB per VLAN multicast packet count low. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.48 VEB VLAN Multicast Packet Count High - GL_STAT_SWR_MPCH[n] (0x00347404 + 0x8*n, n=0...127; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VLMPCH	7:0	0x0	RWC	UNDEFINED	**VLAN Multicast Packet Count High** VEB per VLAN multicast packet count high. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.49 VEB VLAN Broadcast Packet Count Low - GL_STAT_SWR_BPCL[n] (0x00347800 + 0x8*n, n=0...127; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VLBPCL	31:0	0x0	RWC	UNDEFINED	**VLAN Broadcast Packet Count Low** VEB per VLAN broadcast packet count low. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.50 VEB VLAN Broadcast Packet Count High - GL_STAT_SWR_BPCH[n] (0x00347804 + 0x8*n, n=0...127; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VLBPCH	7:0	0x0	RWC	UNDEFINED	**VLAN Broadcast Packet Count High** VEB per VLAN broadcast packet count high. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.51 Port Good Octets Received Count Low - GLPRT_GORCL[n] (0x00380000 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GORCL	31:0	0x0	RWC	UNDEFINED	**Good Octets Received Count Low** Receive octet count. Counts the number of bytes received by this port. Lower 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.52 Port Good Octets Received Count High - GLPRT_GORCH[n] (0x00380004 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GORCH	7:0	0x0	RWC	UNDEFINED	**Good Octets Received Count High** Receive octet count. Counts the number of bytes received by this port. Higher 8 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.53 Port MAC Local Fault Count - GLPRT_MLFC[n] (0x00380040 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MLFC	31:0	0x0	RWC	UNDEFINED	**MAC Local Fault Count** Number of faults in the local MAC.

13.2.2.23.54 Port MAC Local Fault Count - GLPRT_MLFC_H[n] (0x00380044 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MLFC	31:0	0x0	RWC	UNDEFINED	**MAC Local Fault Count** Number of faults in the local MAC.

13.2.2.23.55 Port MAC Remote Fault Count - GLPRT_MRFC[n] (0x00380080 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MRFC	31:0	0x0	RWC	UNDEFINED	**MAC Remote Fault Count** Number of faults in the remote MAC.

13.2.2.23.56 Port MAC Remote Fault Count - GLPRT_MRFC_H[n] (0x00380084 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MRFC	31:0	0x0	RWC	UNDEFINED	**MAC Remote Fault Count** Number of faults in the remote MAC.

13.2.2.23.57 Port CRC Error Count - GLPRT_CRCERRS[n] (0x00380100 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CRCERRS	31:0	0x0	RWC	UNDEFINED	**CRC Errors** CRC error count. Counts the number of receive packets with CRC errors.

13.2.2.23.58 Port CRC Error Count - GLPRT_CRCERRS_H[n] (0x00380104 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CRCERRS	31:0	0x0	RWC	UNDEFINED	**CRC Errors** CRC error count. Counts the number of receive packets with CRC errors.

13.2.2.23.59 Receive Length Error Count - GLPRT_RLEC[n] (0x00380140 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RLEC	31:0	0x0	RWC	UNDEFINED	**Receive Length Error Count** Number of SNAP packets with receive length errors. A length error occurs if an incoming packet length field in the MAC header does not match the packet length. Does not count length error in tagged packets (like VLAN tagged) or non-SNAP packets

13.2.2.23.60 Receive Length Error Count - GLPRT_RLEC_H[n] (0x00380144 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RLEC	31:0	0x0	RWC	UNDEFINED	**Receive Length Error Count** Number of packets with receive length errors. A length error occurs if an incoming packet length field in the MAC header does not match the packet length.

13.2.2.23.61 Port Illegal Byte Error Count - GLPRT_ILLERRC[n] (0x003801C0 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
ILLERRC	31:0	0x0	RWC	UNDEFINED	**Illegal Error Count** Illegal byte error packet count. Counts the number of receive packets with illegal bytes errors (in other words, there is an illegal symbol in the packet).

13.2.2.23.62 Port Illegal Byte Error Count - GLPRT_ILLERRC_H[n] (0x003801C4 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
ILLERRC	31:0	0x0	RWC	UNDEFINED	**Illegal Error Count** Illegal byte error packet count. Counts the number of receive packets with illegal bytes errors (in other words, there is an illegal symbol in the packet).

13.2.2.23.63 Receive Undersize Count - GLPRT_RUC[n] (0x00380200 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RUC	31:0	0x0	RWC	UNDEFINED	**Receive Undersize Count** Receive undersize error. 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.

13.2.2.23.64 Receive Undersize Count - GLPRT_RUC_H[n] (0x00380204 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RUC	31:0	0x0	RWC	UNDEFINED	**Receive Undersize Count** Receive undersize error. 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.

13.2.2.23.65 Receive Oversize Count - GLPRT_ROC[n] (0x00380240 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
ROC	31:0	0x0	RWC	UNDEFINED	**Receive Oversize Count** Receive oversize error. Counts the number of received frames that are longer than maximum size as defined by the Set MAC Config command (from <Destination Address> through <CRC>, inclusively) and have valid CRC.

13.2.2.23.66 Receive Oversize Count - GLPRT_ROC_H[n] (0x00380244 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
ROC	31:0	0x0	RWC	UNDEFINED	**Receive Oversize Count** Receive oversize error. 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.

13.2.2.23.67 Port Link XON Received Count - GLPRT_LXONRXC[n] (0x00380280 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LXONRXCNT	31:0	0x0	RWC	UNDEFINED	**Link XON Received Count** Number of XON packets received.

13.2.2.23.68 Port Link XON Received Count - GLPRT_LXONRXC_H[n] (0x00380284 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LXONRXCNT	31:0	0x0	RWC	UNDEFINED	**Link XON Received Count** Number of XON packets received.

13.2.2.23.69 Port Link XOFF Received Count - GLPRT_LXOFFRXC[n] (0x003802C0 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LXOFFRXCNT	31:0	0x0	RWC	UNDEFINED	**Link XOFF Received Count** Number of XOFF packets received.

13.2.2.23.70 Port Link XOFF Received Count - GLPRT_LXOFFRXC_H[n] (0x003802C4 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LXOFFRXCNT	31:0	0x0	RWC	UNDEFINED	**Link XOFF Received Count** Number of XOFF packets received.

13.2.2.23.71 Priority XON Received Count - GLPRT_PXONRXC[n,m] (0x00380300 + 0x8*n + 0x40*m, n=0...7, m=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRPXONRXCNT	31:0	0x0	RWC	UNDEFINED	**Priority XON Received Count** Number of XON packets received. Array of eight per port.

13.2.2.23.72 Priority XON Received Count - GLPRT_PXONRXC_H[n,m] (0x00380304 + 0x8*n + 0x40*m, n=0...7, m=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRPXONRXCNT	31:0	0x0	RWC	UNDEFINED	**Priority XON Received Count** Number of XON packets received. Array of eight per port.

13.2.2.23.73 Priority XOFF Received Count - GLPRT_PXOFFRXC[n,m] (0x00380500 + 0x8*n + 0x40*m, n=0...7, m=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRPXOFFRXCNT	31:0	0x0	RWC	UNDEFINED	**Priority XOFF Received Count** Number of XOFF packets received. Array of eight per port.

13.2.2.23.74 Priority XOFF Received Count - GLPRT_PXOFFRXC_H[n,m] (0x00380504 + 0x8*n + 0x40*m, n=0...7, m=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRPXOFFRXCNT	31:0	0x0	RWC	UNDEFINED	**Priority XOFF Received Count** Number of XOFF packets received. Array of eight per port.

13.2.2.23.75 Priority XON to XOFF Count - GLPRT_RXON2OFFCNT[n,m] (0x00380700 + 0x8*n + 0x40*m, n=0...7, m=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRRXON2OFFCNT	31:0	0x0	RWC	UNDEFINED	**Priority XON to XOFF Count** Number of times transmitter transitioned from XON to XOFF. Array of eight per port.

13.2.2.23.76 Priority XON to XOFF Count - GLPRT_RXON2OFFCNT_H[n,m] (0x00380704 + 0x8*n + 0x40*m, n=0...7, m=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRRXON2OFFCNT	31:0	0x0	RWC	UNDEFINED	**Priority XON to XOFF Count** Number of times transmitter transitioned from XON to XOFF. Array of eight per port.

13.2.2.23.77 Packets Received [64 Bytes] Count Low - GLPRT_PRC64L[n] (0x00380900 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRC64L	31:0	0x0	RWC	UNDEFINED	**Packets Received Count (64 Bytes) Low** Number of good packets received that are 64 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.78 Packets Received [64 Bytes] Count High - GLPRT_PRC64H[n] (0x00380904 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRC64H	7:0	0x0	RWC	UNDEFINED	**Packets Received Count (64 Bytes) High** Number of good packets received that are 64 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.79 Packets Received [65-127 Bytes] Count Low - GLPRT_PRC127L[n] (0x00380940 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRC127L	31:0	0x0	RWC	UNDEFINED	**Packets Received Count (65-127 Bytes) Low** Number of packets received that are 65-127 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.80 Packets Received [65-127 Bytes] Count High - GLPRT_PRC127H[n] (0x00380944 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRC127H	7:0	0x0	RWC	UNDEFINED	**Packets Received Count (65-127 Bytes) High** Number of packets received that are 65-127 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.81 Packets Received [128-255 Bytes] Count Low - GLPRT_PRC255L[n] (0x00380980 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRC255L	31:0	0x0	RWC	UNDEFINED	**Packets Received Count (128-255 Bytes) Low** Number of packets received that are 128-255 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.82 Packets Received [128-255 Bytes] Count High - GLPRT_PRC255H[n] (0x00380984 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRTPRC255H	7:0	0x0	RWC	UNDEFINED	**Packets Received Count (128-255 Bytes) High** Number of packets received that are 128-255 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.83 Packets Received [256-511 Bytes] Count Low - GLPRT_PRC511L[n] (0x003809C0 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRC511L	31:0	0x0	RWC	UNDEFINED	**Packets Received Count (256-511 Bytes) Low** Number of packets received that are 256-511 bytes in length (from <Destination Address> through <CRC>, inclusively) The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.84 Packets Received [256-511 Bytes] Count High - GLPRT_PRC511H[n] (0x003809C4 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRC511H	7:0	0x0	RWC	UNDEFINED	**Packets Received Count (256-511 Bytes) High** Number of packets received that are 256-511 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.85 Packets Received [512-1023 Bytes] Count Low - GLPRT_PRC1023L[n] (0x00380A00 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRC1023L	31:0	0x0	RWC	UNDEFINED	**Packets Received Count (512-1023 Bytes) Low** Number of packets received that are 512-1023 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.86 Packets Received [512-1023 Bytes] Count High - GLPRT_PRC1023H[n] (0x00380A04 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRC1023H	7:0	0x0	RWC	UNDEFINED	**Packets Received Count (512-1023 Bytes) High** Number of packets received that are 512-1023 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.87 Packets Received [1024-1522 Bytes] Count Low - GLPRT_PRC1522L[n] (0x00380A40 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRC1522L	31:0	0x0	RWC	UNDEFINED	**Packets Received Count (1024-1522 Bytes) Low** Number of packets received that are 1024-1522 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.88 Packets Received [1024-1522 Bytes] Count High - GLPRT_PRC1522H[n] (0x00380A44 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRC1522H	7:0	0x0	RWC	UNDEFINED	**Packets Received Count (1024-1522 Bytes) High** Number of packets received that are 1024-1522 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.89 Packets Received [1523-9522 Bytes] Count Low - GLPRT_PRC9522L[n] (0x00380A80 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRC1522L	31:0	0x0	RWC	UNDEFINED	**Packets Received Count (1523-9522 Bytes) Low** Number of packets received that are 1523-9522 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.90 Packets Received [1523-9522 Bytes] Count High - GLPRT_PRC9522H[n] (0x00380A84 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRC1522H	7:0	0x0	RWC	UNDEFINED	**Packets Received Count (1523-9522 Bytes) High** Number of packets received that are 1523-9522 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.91 Receive Fragment Count - GLPRT_RFC[n] (0x00380AC0 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RFC	31:0	0x0	RWC	UNDEFINED	**Receive Fragments Count** Counts the number of received frames that are shorter than minimum size (64 bytes from <Destination Address> through <CRC>, inclusively) and had an invalid CRC.

13.2.2.23.92 Receive Fragment Count - GLPRT_RFC_H[n] (0x00380AC4 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RFC	31:0	0x0	RWC	UNDEFINED	**Receive Fragments Count** Counts the number of received frames that are shorter than minimum size (64 bytes from <Destination Address> through <CRC>, inclusively) and had an invalid CRC.

13.2.2.23.93 Receive Jabber Count - GLPRT_RJC[n] (0x00380B00 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RJC	31:0	0x0	RWC	UNDEFINED	**Receive Jabber Count** Number of receive jabber errors. Counts the number of received packets that passed address filtering, 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.

13.2.2.23.94 Receive Jabber Count - GLPRT_RJC_H[n] (0x00380B04 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RJC	31:0	0x0	RWC	UNDEFINED	**Receive Jabber Count** Number of receive jabber errors. Counts the number of received packets that passed address filtering, 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.

13.2.2.23.95 Port Good Octets Transmit Count Low - GLPRT_GOTCL[n] (0x00380B40 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GOTCL	31:0	0x0	RWC	UNDEFINED	**Good Octets Transmit Count Low** Transmit octet count. Counts the number of bytes transmitted by this VSI. Lower 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.96 Port Good Octets Transmit Count High - GLPRT_GOTCH[n] (0x00380B44 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GOTCH	7:0	0x0	RWC	UNDEFINED	**Good Octets Transmit Count High** Transmit octet count. Counts the number of bytes transmitted by this port. Higher 8 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.97 Packets Transmitted [64 Bytes] Count Low - GLPRT_PTC64L[n] (0x00380B80 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PTC64L	31:0	0x0	RWC	UNDEFINED	**Packets Transmitted Count (64 Bytes) Low** Number of packets transmitted that are 64 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.98 Packets Transmitted [64 Bytes] Count High - GLPRT_PTC64H[n] (0x00380B84 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PTC64H	7:0	0x0	RWC	UNDEFINED	**Packets Transmitted Count (64 Bytes) High** Number of packets transmitted that are 64 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.99 Packets Transmitted [65-127 Bytes] Count Low - GLPRT_PTC127L[n] (0x00380BC0 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PTC127L	31:0	0x0	RWC	UNDEFINED	**Packets Transmitted Count (65-127 Bytes) Low** Number of packets transmitted that are 65-127 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.100 Packets Transmitted [65-127 Bytes] Count High - GLPRT_PTC127H[n] (0x00380BC4 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PTC127H	7:0	0x0	RWC	UNDEFINED	**Packets Transmitted Count (65-127 Bytes) High** Number of packets transmitted that are 65-127 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.101 Packets Transmitted [128-255 Bytes] Count Low - GLPRT_PTC255L[n] (0x00380C00 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PTC255L	31:0	0x0	RWC	UNDEFINED	**Packets Transmitted Count (128-255 Bytes) Low** Number of packets transmitted that are 128-255 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.102 Packets Transmitted [128-255 Bytes] Count High - GLPRT_PTC255H[n] (0x00380C04 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PTC255H	7:0	0x0	RWC	UNDEFINED	**Packets Transmitted Count (128-255 Bytes) High** Number of packets transmitted that are 128-255 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.103 Packets Transmitted [256-511 Bytes] Count Low - GLPRT_PTC511L[n] (0x00380C40 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PTC511L	31:0	0x0	RWC	UNDEFINED	**Packets Transmitted Count (256-511 Bytes) Low** Number of packets transmitted that are 256-511 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.104 Packets Transmitted [256-511 Bytes] Count High - GLPRT_PTC511H[n] (0x00380C44 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PTC511H	7:0	0x0	RWC	UNDEFINED	**Packets Transmitted Count (256-511 Bytes) High** Number of packets transmitted that are 256-511 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.105 Packets Transmitted [512-1023 Bytes] Count Low - GLPRT_PTC1023L[n] (0x00380C80 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PTC1023L	31:0	0x0	RWC	UNDEFINED	**Packets Transmitted Count (512-1023 Bytes) Low** Number of packets transmitted that are 512-1023 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.106 Packets Transmitted [512-1023 Bytes] Count High - GLPRT_PTC1023H[n] (0x00380C84 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PTC1023H	7:0	0x0	RWC	UNDEFINED	**Packets Transmitted Count (512-1023 Bytes) High** Number of packets transmitted that are 512-1023 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.107 Packets Transmitted [1024-1522 Bytes] Count Low - GLPRT_PTC1522L[n] (0x00380CC0 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PTC1522L	31:0	0x0	RWC	UNDEFINED	**Packets Transmitted Count (1024-1522 Bytes) Low** Number of packets transmitted that are 1024-1522 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.108 Packets Transmitted [1024-1522 Bytes] Count High - GLPRT_PTC1522H[n] (0x00380CC4 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PTC1522H	7:0	0x0	RWC	UNDEFINED	**Packets Transmitted Count (1024-1522 Bytes) High** Number of packets transmitted that are 1024-1522 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.109 Packets Transmitted [1523-9522 bytes] Count Low - GLPRT_PTC9522L[n] (0x00380D00 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PTC9522L	31:0	0x0	RWC	UNDEFINED	**Packets Transmitted Count (1523-9522 Bytes) Low** Number of packets transmitted that are 1523-9522 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.110 Packets Transmitted [1523-9522 bytes] Count High - GLPRT_PTC9522H[n] (0x00380D04 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PTC9522H	7:0	0x0	RWC	UNDEFINED	**Packets Transmitted Count (1523-9522 Bytes) High** Number of packets transmitted that are 1523-9522 bytes in length (from <Destination Address> through <CRC>, inclusively). The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.111 Priority XON Transmitted Count - GLPRT_PXONTXC[n,m] (0x00380D40 + 0x8*n + 0x40*m, n=0...7, m=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRPXONTXC	31:0	0x0	RWC	UNDEFINED	**Priority XON Transmitted Count** Number of XON packets transmitted. Array of eight per port.

13.2.2.23.112 Priority XON Transmitted Count - GLPRT_PXONTXC_H[n,m] (0x00380D44 + 0x8*n + 0x40*m, n=0...7, m=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRPXONTXC	31:0	0x0	RWC	UNDEFINED	**Priority XON Transmitted Count** Number of XON packets transmitted. Array of eight per port.

13.2.2.23.113 Priority XOFF Transmitted Count - GLPRT_PXOFFTXC[n,m] (0x00380F40 + 0x8*n + 0x40*m, n=0...7, m=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRPXOFFTXC	31:0	0x0	RWC	UNDEFINED	**Priority XOFF Transmitted Count** Number of XOFF packets transmitted. Array of eight per port.

13.2.2.23.114 Priority XOFF Transmitted Count - GLPRT_PXOFFTXC_H[n,m] (0x00380F44 + 0x8*n + 0x40*m, n=0...7, m=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PRPXOFFTXC	31:0	0x0	RWC	UNDEFINED	**Priority XOFF Transmitted Count** Number of XOFF packets transmitted. Array of eight per port.

13.2.2.23.115 Port Link XON Transmitted Count - GLPRT_LXONTXC[n] (0x00381140 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LXONTXC	31:0	0x0	RWC	UNDEFINED	**Link XON Transmitted Count** Number of XON packets transmitted. Array of eight per port.

13.2.2.23.116 Port Link XON Transmitted Count - GLPRT_LXONTXC_H[n] (0x00381144 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LXONTXC	31:0	0x0	RWC	UNDEFINED	**Link XON Transmitted Count** Number of XON packets transmitted. Array of eight per port.

13.2.2.23.117 Port Link XOFF Transmitted Count - GLPRT_LXOFFTXC[n] (0x00381180 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LXOFFTXC	31:0	0x0	RWC	UNDEFINED	**Link XOFF Transmitted Count** Number of XOFF packets transmitted. Array of eight per port.

13.2.2.23.118 Port Link XOFF Transmitted Count - GLPRT_LXOFFTXC_H[n] (0x00381184 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LXOFFTXC	31:0	0x0	RWC	UNDEFINED	**Link XOFF Transmitted Count** Number of XOFF packets transmitted. Array of eight per port.

13.2.2.23.119 Port Unicast Packets Transmit Count Low - GLPRT_UPTCL[n] (0x003811C0 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VUPTCH	31:0	0x0	RWC	UNDEFINED	**Unicast Packets Transmit Count Low** Transmit unicast packet count. Counts the number of unicast packets transmitted by this port. Lower 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.120 Port Unicast Packets Transmit Count High - GLPRT_UPTCH[n] (0x003811C4 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UPTCH	7:0	0x0	RWC	UNDEFINED	**Unicast Packets Transmit Count High** Transmit unicast packet count. Counts the number of unicast packets transmitted by this port. Higher 8 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.121 Port Multicast Packets Transmit Count Low - GLPRT_MPTCL[n] (0x00381200 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MPTCL	31:0	0x0	RWC	UNDEFINED	**Multicast Packets Transmit Count Low** Transmit multicast packet count. Counts the number of multicast packets transmitted by this port. Lower 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.122 Port Multicast Packets Transmit Count High - GLPRT_MPTCH[n] (0x00381204 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MPTCH	7:0	0x0	RWC	UNDEFINED	**Multicast Packets Transmit Count High** Transmit multicast packet count. Counts the number of multicast packets transmitted by this port. Higher 8 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.123 Port Broadcast Packets Transmit Count Low - GLPRT_BPTCL[n] (0x00381240 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UPRCH	31:0	0x0	RWC	UNDEFINED	**Broadcast Packets Transmit Count Low** Transmit broadcast packet count. Counts the number of broadcast packets transmitted by this port. Lower 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.124 Port Broadcast Packets Transmit Count High - GLPRT_BPTCH[n] (0x00381244 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UPRCH	7:0	0x0	RWC	UNDEFINED	**Broadcast Packets Transmit Count High** Transmit broadcast packet count. Counts the number of broadcast packets transmitted by this port. Higher 8 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.125 Transmit Discard on Link Down - GLPRT_TDOLD[n] (0x00381280 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GLPRT_TDOLD	31:0	0x0	RWC	UNDEFINED	**Transmit Discard on Link Down** Packets discarded at the port because the link was down.

13.2.2.23.126 Transmit Discard on Link Down - GLPRT_TDOLD_H[n] (0x00381284 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GLPRT_TDOLD	31:0	0x0	RWC	UNDEFINED	**Transmit Discard on Link Down** Packets discarded at the port because the link was down.

13.2.2.23.127 Port Unicast Packets Received Count Low - GLPRT_UPRCL[n] (0x00381300 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UPRCL	31:0	0x0	RWC	UNDEFINED	**Unicast Packets Received Count Low** Receive unicast packet count. Counts the number of unicast packets received by this port. Lower 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.128 Port Unicast Packets Received Count High - GLPRT_UPRCH[n] (0x00381304 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UPRCH	7:0	0x0	RWC	UNDEFINED	**Unicast Packets Received Count High** Receive unicast packet count. Counts the number of unicast packets received by this port. Higher 8 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.129 Port Multicast Packets Received Count Low - GLPRT_MPRCL[n] (0x00381340 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MPRCL	31:0	0x0	RWC	UNDEFINED	**Multicast Packets Received Count Low** Receive multicast packet count. Counts the number of multicast packets received by this port. Lower 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.130 Port Multicast Packets Received Count High - GLPRT_MPRCH[n] (0x00381344 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MPRCH	7:0	0x0	RWC	UNDEFINED	**Multicast Packets Received Count High** Receive multicast packet count. Counts the number of multicast packets received by this port. Higher 8 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.131 Port Broadcast packets received count low - GLPRT_BPRCL[n] (0x00381380 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UPRCH	31:0	0x0	RWC	UNDEFINED	**Broadcast Packets Received Count Low** Receive broadcast packet count. Counts the number of broadcast packets received by this port. Lower 32 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.

13.2.2.23.132 Port Broadcast Packets Received Count High - GLPRT_BPRCH[n] (0x00381384 + 0x8*n, n=0...7; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UPRCH	7:0	0x0	RWC	UNDEFINED	**Broadcast Packets Received Count High** Receive broadcast packet count. Counts the number of broadcast packets received by this port. Higher 8 bits. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.133 ACL Counter Bank 0 LSBs - GLSTAT_ACL_CNT_0_L[n] (0x00388000 + 0x8*n, n=0...511; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CNT_LSB	31:0	0x0	RWC	UNDEFINED	**Counter LSB** Packet octet packet byte count or packet count for ACL hits from ACL action - bank0.

13.2.2.23.134 ACL Counter Bank 0 MSBs - GLSTAT_ACL_CNT_0_H[n] (0x00388004 + 0x8*n, n=0...511; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CNT_MSB	7:0	0x0	RWC	UNDEFINED	**Counter MSB** Packet octet packet byte count or packet count for ACL hits from ACL action - bank0.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.135 ACL Counter Bank 1 LSBs - GLSTAT_ACL_CNT_1_L[n] (0x00389000 + 0x8*n, n=0...511; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CNT_LSB	31:0	0x0	RWC	UNDEFINED	**Counter LSB** Packet octet packet byte count or packet count for ACL hits from ACL action - bank1.

13.2.2.23.136 ACL Counter Bank 1 MSBs - GLSTAT_ACL_CNT_1_H[n] (0x00389004 + 0x8*n, n=0...511; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CNT_MSB	7:0	0x0	RWC	UNDEFINED	**Counter MSB** Packet octet packet byte count or packet count for ACL hits from ACL action - bank1.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.137 ACL Counter Bank 2 LSBs - GLSTAT_ACL_CNT_2_L[n] (0x0038A000 + 0x8*n, n=0...511; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CNT_LSB	31:0	0x0	RWC	UNDEFINED	**Counter LSB** Packet octet packet byte count or packet count for ACL hits from ACL action - bank2.

13.2.2.23.138 ACL Counter Bank 2 MSBs - GLSTAT_ACL_CNT_2_H[n] (0x0038A004 + 0x8*n, n=0...511; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CNT_MSB	7:0	0x0	RWC	UNDEFINED	**Counter MSB** Packet octet packet byte count or packet count for ACL hits from ACL action - bank2.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.139 ACL Counter Bank 3 LSBs - GLSTAT_ACL_CNT_3_L[n] (0x0038B000 + 0x8*n, n=0...511; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CNT_LSB	31:0	0x0	RWC	UNDEFINED	**Counter LSB** Packet octet packet byte count or packet count for ACL hits from ACL action - bank3.

13.2.2.23.140 ACL Counter Bank 3 MSBs - GLSTAT_ACL_CNT_3_H[n] (0x0038B004 + 0x8*n, n=0...511; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CNT_MSB	7:0	0x0	RWC	UNDEFINED	**Counter MSB** Packet octet packet byte count or packet count for ACL hits from ACL action - bank3.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.141 Global Packet Byte Statistic Counter Bank 0 Low - GLSTAT_FD_CNT0L[n] (0x003A0000 + 0x8*n, n=0...4095; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FD0_CNT_L	31:0	0x0	RWC	UNDEFINED	**FD0 Counter Low** Low 32 bits of packet/byte counter referenced by FD filters. The low and high registers are part of a 64-bit register and can be read using 64-bit read accesses. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.23.142 Global Packet Byte Statistic Counter Bank 0 High - GLSTAT_FD_CNT0H[n] (0x003A0004 + 0x8*n, n=0...4095; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FD0_CNT_H	7:0	0x0	RWC	UNDEFINED	**FD0 Counter High** High 8 bits of packet/byte counter referenced by FD filters. The low and high registers are part of a 64-bit register and can be read using 64-bit read accesses. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.143 Global Packet Byte Statistic Counter Bank 1 Low - GLSTAT_FD_CNT1L[n] (0x003A8000 + 0x8*n, n=0...4095; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FD0_CNT_L	31:0	0x0	RWC	UNDEFINED	**FD1 Counter Low** Low 32 bits of packet/byte counter referenced by FD filters. The low and high registers are part of a 64-bit register and can be read using 64-bit read accesses. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.23.144 Global Packet Byte Statistic Counter Bank 1 High - GLSTAT_FD_CNT1H[n] (0x003A8004 + 0x8*n, n=0...4095; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FD0_CNT_H	7:0	0x0	RWC	UNDEFINED	**FD1 Counter High** High 8 bits of packet/byte counter referenced by FD filters. The low and high registers are part of a 64-bit register and can be read using 64-bit read accesses. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.145 VSI Good Octets Received Count Low - GLV_GORCL[n] (0x003B0000 + 0x8*n, n=0...767; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GORCL	31:0	0x0	RWC	UNDEFINED	**Good Octets Received Count Low** Receive octet count. Counts the number of bytes received by this VSI. Lower 32 bits. The low and high registers are part of a 64-bit register and can be read using 64-bit read accesses. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.23.146 VSI Good Octets Received Count High - GLV_GORCH[n] (0x003B0004 + 0x8*n, n=0...767; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GORCH	7:0	0x0	RWC	UNDEFINED	**Good Octets Received Count High** Receive octet count. Counts the number of bytes received by this VSI. Higher 8 bits. The low and high registers are part of a 64-bit register and can be read using 64-bit read accesses. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.147 VSI Unicast Packets Received Count Low - GLV_UPRCL[n] (0x003B2000 + 0x8*n, n=0...767; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UPRCL	31:0	0x0	RWC	UNDEFINED	**Unicast Packets Received Count Low** Receive unicast packet count. Counts the number of unicast packets received by this VSI. Lower 32 bits. The low and high registers are part of a 64-bit register and can be read using 64-bit read accesses. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.23.148 VSI Unicast Packets Received Count High - GLV_UPRCH[n] (0x003B2004 + 0x8*n, n=0...767; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UPRCH	7:0	0x0	RWC	UNDEFINED	**Unicast Packets Received Count High** Receive unicast packet count. Counts the number of unicast packets received by this VSI. Higher 8 bits. The low and high registers are part of a 64-bit register and can be read using 64-bit read accesses. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.149 VSI Multicast Packets Received Count Low - GLV_MPRCL[n] (0x003B4000 + 0x8*n, n=0...767; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MPRCL	31:0	0x0	RWC	UNDEFINED	**Multicast Packets Received Count Low** Receive multicast packet count. Counts the number of multicast packets received by this VSI. Lower 32 bits. The low and high registers are part of a 64-bit register and can be read using 64-bit read accesses. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.23.150 VSI Multicast Packets Received Count High - GLV_MPRCH[n] (0x003B4004 + 0x8*n, n=0...767; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MPRCH	7:0	0x0	RWC	UNDEFINED	**Multicast Packets Received Count High** Receive multicast packet count. Counts the number of multicast packets received by this VSI. Higher 8 bits. The low and high registers are part of a 64-bit register and can be read using 64-bit read accesses. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.23.151 VSI Broadcast Packets Received Count Low - GLV_BPRCL[n] (0x003B6000 + 0x8*n, n=0...767; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BPRCL	31:0	0x0	RWC	UNDEFINED	**Broadcast Packets Received Count Low** Receive broadcast packet count. Counts the number of broadcast packets received by this VSI. Lower 32 bits. The low and high registers are part of a 64-bit register and can be read using 64-bit read accesses. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.23.152 VSI Broadcast Packets Received Count High - GLV_BPRCH[n] (0x003B6004 + 0x8*n, n=0...767; RWC)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BPRCH	7:0	0x0	RWC	UNDEFINED	**Broadcast Packets Received Count High** Receive broadcast packet count. Counts the number of broadcast packets received by this VSI. Higher 8 bits. The low and high registers are part of a 64-bit register and can be read using 64-bit read accesses. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.24 PF - Protocol Engine Statistics Registers

13.2.2.24.1 Protocol Engine Statistics Received VLAN_ID Errors - GLPES_PFRXVLANERR[n] (0x00540000 + 0x4*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RXVLANERR	23:0	0x0	RW1C	DYNAMIC	**Received VLAN_ID Errors** Counts the number of packets received by the Protocol Engine with incorrect VLAN_ID.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.24.2 Protocol Engine Statistics IPv4 Received Octets Low - GLPES_PFIP4RXOCTSLO[n] (0x00540400 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4RXOCTSLO	31:0	0x0	RW1C	DYNAMIC	**IPv4 Received Octets Low** Counts the number of IPv4 octets received by the Protocol Engine. This is the low 32 bits of the 48-bit counter. This counter is not incremented for received IPv4 multicast packets. Software must add a value of this counter to the IPv4 Multicast Octet counter to calculate a total number of octets received. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.3 Protocol Engine Statistics IPv4 Received Octets High - GLPES_PFIP4RXOCTSHI[n] (0x00540404 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4RXOCTSHI	15:0	0x0	RW1C	DYNAMIC	**IPv4 Received Octets High** Counts the number of IPv4 octets received by the Protocol Engine. This is the high 16 bits of the 48-bit counter. This counter is not incremented for received IPv4 multicast packets. Software must add a value of this counter to the IPv4 Multicast Octet counter to calculate a total number of octets received. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.4 Protocol Engine Statistics IPv4 Received Packets Low - GLPES_PFIP4RXPKTSLO[n] (0x00540C00 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4RXPKTSLO	31:0	0x0	RW1C	DYNAMIC	**IPv4 Received Packets Low** Counts the number of IPv4 packets received by the Protocol Engine. This is the low 32 bits of the 48-bit counter. This counter is not incremented for received IPv4 multicast packets. Software must add a value of this counter to the IPv4 Multicast Packet counter to calculate a total number of packets received. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.5 Protocol Engine Statistics IPv4 Received Packets High - GLPES_PFIP4RXPKTSHI[n] (0x00540C04 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4RXPKTSHI	15:0	0x0	RW1C	DYNAMIC	**IPv4 Received Packets High** Counts the number of IPv4 packets received by the Protocol Engine. This is the high 16 bits of the 48-bit counter. This counter is not incremented for received IPv4 multicast packets. Software must add a value of this counter to the IPv4 Multicast Packet counter to calculate a total number of packets received. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.6 Protocol Engine Statistics IPv4 Discards - GLPES_PFIP4RXDISCARD[n] (0x00541400 + 0x4*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4RXDISCARD	31:0	0x0	RW1C	DYNAMIC	**IPv4 Received Discarded** Counts the number of IPv4 packets received by the Protocol Engine without errors and discarded.

13.2.2.24.7 Protocol Engine Statistics IPv4 Truncated Packets - GLPES_PFIP4RXTRUNC[n] (0x00541800 + 0x4*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4RXTRUNC	31:0	0x0	RW1C	DYNAMIC	**IPv4 Received Truncated** Counts the number of IPv4 packets received by the Protocol Engine and truncated due to insufficient payload or header buffering space in RQ descriptors.

13.2.2.24.8 Protocol Engine Statistics IPv4 Received Fragments Low - GLPES_PFIP4RXFRAGSLO[n] (0x00541C00 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4RXFRAGSLO	31:0	0x0	RW1C	DYNAMIC	**IPv4 Received Fragments Low** Counts the number of IPv4 fragments received by the Protocol Engine. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.9 Protocol Engine Statistics IPv4 Received Fragments High - GLPES_PFIP4RXFRAGSHI[n] (0x00541C04 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4RXFRAGSHI	15:0	0x0	RW1C	DYNAMIC	**IPv4 Received Fragments High** Counts the number of IPv4 fragments received by the Protocol Engine. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.10 Protocol Engine Statistics IPv4 Received Multicast Octets Low - GLPES_PFIP4RXMCOCTSLO[n] (0x00542400 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4RXMCOCTSLO	31:0	0x0	RW1C	DYNAMIC	**IPv4 Received Multicast Octets Low** Counts the number of IPv4 multicast octets received by the Protocol Engine. This is the low 32 bits of the 48-bit counter. This counter does not count number of octets of the multicast packets replicated inside Protocol Engine. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.11 Protocol Engine Statistics IPv4 Received Multicast Octets High - GLPES_PFIP4RXMCOCTSHI[n] (0x00542404 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4RXMCOCTSHI	15:0	0x0	RW1C	DYNAMIC	**IPv4 Received Multicast Octets High** Counts the number of IPv4 multicast octets received by the Protocol Engine. This is the high 16 bits of the 48-bit counter. This counter does not count number of octets of the multicast packets replicated inside Protocol Engine. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.12 Protocol Engine Statistics IPv4 Received Multicast Packets Low - GLPES_PFIP4RXMCPKTSLO[n] (0x00542C00 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4RXMCPKTSLO	31:0	0x0	RW1C	DYNAMIC	**IPv4 Received Multicast Packets Low** Counts the number of IPv4 multicast packets received by the Protocol Engine. This is the low 32 bits of the 48-bit counter. This counter does not count number of multicast packets replicated inside Protocol Engine. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.13 Protocol Engine Statistics IPv4 Received Multicast Packets High - GLPES_PFIP4RXMCPKTSHI[n] (0x00542C04 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4RXMCPKTSHI	15:0	0x0	RW1C	DYNAMIC	**IPv4 Received Multicast Packets High** Counts the number of IPv4 multicast packets received by the Protocol Engine. This is the high 16 bits of the 48-bit counter. This counter does not count number of multicast packets replicated inside Protocol Engine. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.14 Protocol Engine Statistics IPv6 Received Octets Low - GLPES_PFIP6RXOCTSLO[n] (0x00543400 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6RXOCTSLO	31:0	0x0	RW1C	DYNAMIC	**Pv6 Received Octets Low** Counts the number of IPv6 octets received by the Protocol Engine. This is the low 32 bits of the 48-bit counter. This counter is not incremented for received IPv6 multicast packets. Software must add a value of this counter to the IPv6 Multicast Octet counter to calculate a total number of octets received. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.15 Protocol Engine Statistics IPv6 Received Octets High - GLPES_PFIP6RXOCTSHI[n] (0x00543404 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6RXOCTSHI	15:0	0x0	RW1C	DYNAMIC	**Pv6 Received Octets High** Counts the number of IPv6 octets received by the Protocol Engine. This is the high 16 bits of the 48-bit counter. This counter is not incremented for received IPv6 multicast packets. Software must add a value of this counter to the IPv6 Multicast Octet counter to calculate a total number of octets received. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.16 Protocol Engine Statistics IPv6 Received Packets Low - GLPES_PFIP6RXPKTSLO[n] (0x00543C00 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6RXPKTSLO	31:0	0x0	RW1C	DYNAMIC	**IPv6 Received Packets Low** Counts the number of IPv6 packets received by the Protocol Engine. This is the low 32 bits of the 48-bit counter. This counter is not incremented for received IPv6 multicast packets. Software must add a value of this counter to the IPv6 Multicast Packet counter to calculate a total number of packets received. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.17 Protocol Engine Statistics IPv6 Received Packets High - GLPES_PFIP6RXPKTSHI[n] (0x00543C04 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6RXPKTSHI	15:0	0x0	RW1C	DYNAMIC	**IPv6 Received Packets High** Counts the number of IPv6 packets received by the Protocol Engine. This is the high 16 bits of the 48-bit counter. This counter is not incremented for received IPv6 multicast packets. Software must add a value of this counter to the IPv6 Multicast Packet counter to calculate a total number of packets received. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.18 Protocol Engine Statistics IPv6 Discards - GLPES_PFIP6RXDISCARD[n] (0x00544400 + 0x4*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6RXDISCARD	31:0	0x0	RW1C	DYNAMIC	**IPv6 Received Discarded** Counts the number of IPv6 packets received by the Protocol Engine without errors and discarded.

13.2.2.24.19 Protocol Engine Statistics IPv6 Truncated Packets - GLPES_PFIP6RXTRUNC[n] (0x00544800 + 0x4*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6RXTRUNC	31:0	0x0	RW1C	DYNAMIC	**IPv6 Received Truncated** Counts the number of IPv6 packets received by the Protocol Engine and truncated due to insufficient payload or header buffering space in RQ descriptors.

13.2.2.24.20 Protocol Engine Statistics IPv6 Received Fragments Low - GLPES_PFIP6RXFRAGSLO[n] (0x00544C00 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6RXFRAGSLO	31:0	0x0	RW1C	DYNAMIC	**IPv6 Received Fragments Low** Counts the number of IPv6 fragments received by the Protocol Engine. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.21 Protocol Engine Statistics IPv6 Received Fragments High - GLPES_PFIP6RXFRAGSHI[n] (0x00544C04 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6RXFRAGSHI	15:0	0x0	RW1C	DYNAMIC	**IPv6 Received Fragments High** Counts the number of IPv6 fragments received by the Protocol Engine. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.22 Protocol Engine Statistics IPv6 Received Multicast Octets Low - GLPES_PFIP6RXMCOCTSLO[n] (0x00545400 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6RXMCOCTSLO	31:0	0x0	RW1C	DYNAMIC	**IPv6 Received Multicast Octets Low** Counts the number of IPv6 multicast octets received by the Protocol Engine. This is the low 32 bits of the 48-bit counter. This counter does not count number of octets of the multicast packets replicated inside Protocol Engine. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.23 Protocol Engine Statistics IPv6 Received Multicast Octets High - GLPES_PFIP6RXMCOCTSHI[n] (0x00545404 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6RXMCPKTSHI	15:0	0x0	RW1C	DYNAMIC	**IPv6 Received Multicast Packets High** Counts the number of IPv6 multicast packets received by the Protocol Engine. This is the high 16 bits of the 48-bit counter. This counter does not count number of multicast packets replicated inside Protocol Engine. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.24 Protocol Engine Statistics IPv6 Received Multicast Packets Low - GLPES_PFIP6RXMCPKTSLO[n] (0x00545C00 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6RXMCPKTSLO	31:0	0x0	RW1C	DYNAMIC	**IPv6 Received Multicast Packets Low** Counts the number of IPv6 multicast packets received by the Protocol Engine. This is the low 32 bits of the 48-bit counter. This counter does not count number of multicast packets replicated inside Protocol Engine. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.25 Protocol Engine Statistics IPv6 Received Multicast Packets High - GLPES_PFIP6RXMCPKTSHI[n] (0x00545C04 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6RXMCPKTSHI	15:0	0x0	RW1C	DYNAMIC	**IPv6 Received Multicast Packets High** Counts the number of IPv6 multicast packets received by the Protocol Engine. This is the high 16 bits of the 48-bit counter. This counter does not count number of multicast packets replicated inside Protocol Engine. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.26 Protocol Engine Statistics IPv4 Transmitted Octets Low - GLPES_PFIP4TXOCTSLO[n] (0x00546400 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4TXOCTSLO	31:0	0x0	RW1C	DYNAMIC	**IPv4 Transmitted Octets Low** Counts the number of IPv4 octets supplied by the Protocol Engine to the lower layers for transmission. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.27 Protocol Engine Statistics IPv4 Transmitted Octets High - GLPES_PFIP4TXOCTSHI[n] (0x00546404 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4TXOCTSHI	15:0	0x0	RW1C	DYNAMIC	**IPv4 Transmitted Octets High** Counts the number of IPv4 octets supplied by the Protocol Engine to the lower layers for transmission. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.28 Protocol Engine Statistics IPv4 Transmitted Packets Low - GLPES_PFIP4TXPKTSLO[n] (0x00546C00 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4TXPKTSLO	31:0	0x0	RW1C	DYNAMIC	**IPv4 Transmitted Packets Low** Counts the number of IPv4 packets supplied by the Protocol Engine to the lower layers for transmission. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.29 Protocol Engine Statistics IPv4 Transmitted Packets High - GLPES_PFIP4TXPKTSHI[n] (0x00546C04 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4TXPKTSHI	15:0	0x0	RW1C	DYNAMIC	**IPv4 Transmitted Packets High** Counts the number of IPv4 packets supplied by the Protocol Engine to the lower layers for transmission. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.30 Protocol Engine Statistics IPv4 Transmitted Fragments Low - GLPES_PFIP4TXFRAGSLO[n] (0x00547400 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4TXFRAGSLO	31:0	0x0	RW1C	DYNAMIC	**IPv4 Transmitted Fragments Low** Counts the number of IPv4 fragments supplied by the Protocol Engine to the lower layers for transmission. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.31 Protocol Engine Statistics IPv4 Transmitted Fragments High - GLPES_PFIP4TXFRAGSHI[n] (0x00547404 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4TXFRAGSHI	15:0	0x0	RW1C	DYNAMIC	**IPv4 Transmitted Fragments High** Counts the number of IPv4 fragments supplied by the Protocol Engine to the lower layers for transmission. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.32 Protocol Engine Statistics IPv4 Transmitted Multicast Octets Low - GLPES_PFIP4TXMCOCTSLO[n] (0x00547C00 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4TXMCOCTSLO	31:0	0x0	RW1C	DYNAMIC	**IPv4 Transmitted Multicast Octets Low** Counts the number of IPv4 multicast octets supplied by the Protocol Engine to the lower layers for transmission. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.33 Protocol Engine Statistics IPv4 Transmitted Multicast Octets High - GLPES_PFIP4TXMCOCTSHI[n] (0x00547C04 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4TXMCOCTSHI	15:0	0x0	RW1C	DYNAMIC	**IPv4 Transmitted Multicast Octets High** Counts the number of IPv4 multicast octets supplied by the Protocol Engine to the lower layers for transmission. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.34 Protocol Engine Statistics IPv4 Transmitted Multicast Packets Low - GLPES_PFIP4TXMCPKTSLO[n] (0x00548400 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4TXMCPKTSLO	31:0	0x0	RW1C	DYNAMIC	**IPv4 Transmitted Multicast Packets Low** Counts the number of IPv4 multicast packets supplied by the Protocol Engine to the lower layers for transmission. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.35 Protocol Engine Statistics IPv4 Transmitted Multicast Packets High - GLPES_PFIP4TXMCPKTSHI[n] (0x00548404 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4TXMCPKTSHI	15:0	0x0	RW1C	DYNAMIC	**IPv4 Transmitted Multicast Packets High** Counts the number of IPv4 multicast packets supplied by the Protocol Engine to the lower layers for transmission. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.36 Protocol Engine Statistics IPv6 Transmitted Octets Low - GLPES_PFIP6TXOCTSLO[n] (0x00548C00 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6TXOCTSLO	31:0	0x0	RW1C	DYNAMIC	**IPv6 Transmitted Octets Low** Counts the number of IPv6 octets supplied by the Protocol Engine to the lower layers for transmission. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.37 Protocol Engine Statistics IPv6 Transmitted Octets High - GLPES_PFIP6TXOCTSHI[n] (0x00548C04 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6TXOCTSHI	15:0	0x0	RW1C	DYNAMIC	**IPv6 Transmitted Octets High** Counts the number of IPv6 octets supplied by the Protocol Engine to the lower layers for transmission. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.38 Protocol Engine Statistics IPv6 Transmitted Packets Low - GLPES_PFIP6TXPKTSLO[n] (0x00549400 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6TXPKTSLO	31:0	0x0	RW1C	DYNAMIC	**IPv6 Transmitted Packets Low** Counts the number of IPv6 packets supplied by the Protocol Engine to the lower layers for transmission. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.39 Protocol Engine Statistics IPv6 Transmitted Packets High - GLPES_PFIP6TXPKTSHI[n] (0x00549404 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6TXPKTSHI	15:0	0x0	RW1C	DYNAMIC	**IPv6 Transmitted Packets High** Counts the number of IPv6 packets supplied by the Protocol Engine to the lower layers for transmission. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.40 Protocol Engine Statistics IPv6 Transmitted Fragments Low - GLPES_PFIP6TXFRAGSLO[n] (0x00549C00 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6TXFRAGSLO	31:0	0x0	RW1C	DYNAMIC	**IPv6 Transmitted Fragments Low** Counts the number of IPv6 fragments supplied by the Protocol Engine to the lower layers for transmission. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.41 Protocol Engine Statistics IPv6 Transmitted Fragments High - GLPES_PFIP6TXFRAGSHI[n] (0x00549C04 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6TXFRAGSHI	15:0	0x0	RW1C	DYNAMIC	**IPv6 Transmitted Fragments High** Counts the number of IPv6 fragments supplied by the Protocol Engine to the lower layers for transmission. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.42 Protocol Engine Statistics IPv6 Transmitted Multicast Octets Low - GLPES_PFIP6TXMCOCTSLO[n] (0x0054A400 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6TXMCOCTSLO	31:0	0x0	RW1C	DYNAMIC	**IPv6 Transmitted Multicast Octets Low** Counts the number of IPv6 multicast octets supplied by the Protocol Engine to the lower layers for transmission. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.43 Protocol Engine Statistics IPv6 Transmitted Multicast Octets High - GLPES_PFIP6TXMCOCTSHI[n] (0x0054A404 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6TXMCOCTSHI	15:0	0x0	RW1C	DYNAMIC	**IPv6 Transmitted Multicast Octets High** Counts the number of IPv6 multicast octets supplied by the Protocol Engine to the lower layers for transmission. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.44 Protocol Engine Statistics IPv6 Transmitted Multicast Packets Low - GLPES_PFIP6TXMCPKTSLO[n] (0x0054AC00 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6TXMCPKTSLO	31:0	0x0	RW1C	DYNAMIC	**IPv6 Transmitted Multicast Packets Low** Counts the number of IPv6 multicast packets supplied by the Protocol Engine to the lower layers for transmission. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.45 Protocol Engine Statistics IPv6 Transmitted Multicast Packets High - GLPES_PFIP6TXMCPKTSHI[n] (0x0054AC04 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6TXMCPKTSHI	15:0	0x0	RW1C	DYNAMIC	**IPv6 Transmitted Multicast Packets High** Counts the number of IPv6 multicast packets supplied by the Protocol Engine to the lower layers for transmission. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.46 Protocol Engine Statistics IPv4 Discarded No Route Packets - GLPES_PFIP4TXNOROUTE[n] (0x0054B400 + 0x4*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP4TXNOROUTE	23:0	0x0	RW1C	DYNAMIC	**IPv4 Transmitted No Route** Counts the number of IPv4 packets discarded due to routing problem (no hit in ARP table).
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.24.47 Protocol Engine Statistics IPv6 Discarded No Route Packets - GLPES_PFIP6TXNOROUTE[n] (0x0054B800 + 0x4*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IP6TXNOROUTE	23:0	0x0	RW1C	DYNAMIC	**IPv6 Transmitted No Route** Counts the number of IPv6 packets discarded due to routing problem (no hit in ARP table).
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.24.48 Protocol Engine Statistics TCP Received Segments Low - GLPES_PFTCPRXSEGSLO[n] (0x0054BC00 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPRXSEGSLO	31:0	0x0	RW1C	DYNAMIC	**TCP Received Segments Low** Counts the number of TCP segments received by the Protocol Engine. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.49 Protocol Engine Statistics TCP Received Segments High - GLPES_PFTCPRXSEGSHI[n] (0x0054BC04 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPRXSEGSHI	15:0	0x0	RW1C	DYNAMIC	**TCP Received Segments High** Counts the number of TCP segments received by the Protocol Engine. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.50 Protocol Engine Statistics TCP Received Segments with Unsupported Options - GLPES_PFTCPRXOPTERR[n] (0x0054C400 + 0x4*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPRXOPTERR	23:0	0x0	RW1C	DYNAMIC	**TCP Received Options Errors** Counts the number of TCP segments received by the Protocol Engine with unsupported TCP options and TCP option length errors.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.24.51 Protocol Engine Statistics TCP Dropped Segments due Protocol Errors - GLPES_PFTCPRXPROTOERR[n] (0x0054C800 + 0x4*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPRXPROTOERR	23:0	0x0	RW1C	DYNAMIC	**TCP Received Protocol Errors** Counts the number of TCP segments received and dropped by the Protocol Engine due to TCP protocol errors.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.24.52 Protocol Engine Statistics TCP Transmitted Segments Low - GLPES_PFTCPTXSEGLO[n] (0x0054CC00 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPTXSEGLO	31:0	0x0	RW1C	DYNAMIC	**TCP Transmitted Segments Low** Counts the number of TCP segments supplied by the Protocol Engine to the lower layers for transmission. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.53 Protocol Engine Statistics TCP Transmitted Segments High - GLPES_PFTCPTXSEGHI[n] (0x0054CC04 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPTXSEGHI	15:0	0x0	RW1C	DYNAMIC	**TCP Transmitted Segments High** Counts the number of TCP segments supplied by the Protocol Engine to the lower layers for transmission. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.54 Protocol Engine Statistics UDP Received Packets Low - GLPES_PFUDPRXPKTSLO[n] (0x0054D400 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UDPRXPKTSLO	31:0	0x0	RW1C	DYNAMIC	**UDP Received Packets Low** Counts the number of UDP packets received by the Protocol Engine without errors. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.55 Protocol Engine Statistics UDP Received Packets High - GLPES_PFUDPRXPKTSHI[n] (0x0054D404 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UDPRXPKTSHI	15:0	0x0	RW1C	DYNAMIC	**UDP Received Packets High** Counts the number of UDP packets received by the Protocol Engine without errors. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.56 Protocol Engine Statistics UDP Transmitted Packets Low - GLPES_PFUDPTXPKTSLO[n] (0x0054DC00 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UDPTXPKTSLO	31:0	0x0	RW1C	DYNAMIC	**UDP Transmitted Packets Low** Counts the number of UDP packets submitted by the Protocol Engine to the lower layers for transmission. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.57 Protocol Engine Statistics UDP Transmitted Packets High - GLPES_PFUDPTXPKTSHI[n] (0x0054DC04 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
UDPTXPKTSHI	15:0	0x0	RW1C	DYNAMIC	**UDP Transmitted Packets High** Counts the number of UDP packets submitted by the Protocol Engine to the lower layers for transmission. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.58 Protocol Engine Statistics RDMA Received Write Messages Low - GLPES_PFRDMARXWRSLO[n] (0x0054E400 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMARXWRSLO	31:0	0x0	RW1C	DYNAMIC	**RDMA Received Writes Low** Counts the number of RDMA Write messages received by the Protocol Engine. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.59 Protocol Engine Statistics RDMA Received Write Messages High - GLPES_PFRDMARXWRSHI[n] (0x0054E404 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMARXWRSHI	15:0	0x0	RW1C	DYNAMIC	**RDMA Received Writes High** Counts the number of RDMA Write messages received by the Protocol Engine. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.60 Protocol Engine Statistics RDMA Received Read Request Messages Low - GLPES_PFRDMARXRDSLO[n] (0x0054EC00 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMARXRDSLO	31:0	0x0	RW1C	DYNAMIC	**RDMA Received Reads Low** Counts the number of RDMA Read Request messages received by the Protocol Engine. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.61 Protocol Engine Statistics RDMA Received Read Request Messages High - GLPES_PFRDMARXRDSHI[n] (0x0054EC04 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMARXRDSHI	15:0	0x0	RW1C	DYNAMIC	**RDMA Received Reads High** Counts the number of RDMA Read Request messages received by the Protocol Engine. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.62 Protocol Engine Statistics RDMA Received Send Messages Low - GLPES_PFRDMARXSNDSLO[n] (0x0054F400 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMARXSNDSLO	31:0	0x0	RW1C	DYNAMIC	**RDMA Received Sends Low** Counts the number of RDMA Send messages received by the Protocol Engine. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.63 Protocol Engine Statistics RDMA Received Send Messages High - GLPES_PFRDMARXSNDSHI[n] (0x0054F404 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMARXSNDSHI	15:0	0x0	RW1C	DYNAMIC	**RDMA Received Sends High** Counts the number of RDMA Send messages received by the Protocol Engine. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.64 Protocol Engine Statistics RDMA Transmitted Write Messages Low - GLPES_PFRDMATXWRSLO[n] (0x0054FC00 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMARXWRSLO	31:0	0x0	RW1C	DYNAMIC	**RDMA Transmitted Writes Low** Counts the number of RDMA Write messages transmitted by the Protocol Engine. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.65 Protocol Engine Statistics RDMA Transmitted Write Messages High - GLPES_PFRDMATXWRSHI[n] (0x0054FC04 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMARXWRSHI	15:0	0x0	RW1C	DYNAMIC	**RDMA Transmitted Writes High** Counts the number of RDMA Write messages transmitted by the Protocol Engine. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.66 Protocol Engine Statistics RDMA Transmitted Read Request Messages Low - GLPES_PFRDMATXRDSLO[n] (0x00550400 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMARXRDSLO	31:0	0x0	RW1C	DYNAMIC	**RDMA Transmitted Reads Low** Counts the number of RDMA Read Request messages transmitted by the Protocol Engine. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.67 Protocol Engine Statistics RDMA Transmitted Read Request Messages High - GLPES_PFRDMATXRDSHI[n] (0x00550404 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMARXRDSHI	15:0	0x0	RW1C	DYNAMIC	**RDMA Transmitted Reads High** Counts the number of RDMA Read Request messages transmitted by the Protocol Engine. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.68 Protocol Engine Statistics RDMA Transmitted Send Messages Low - GLPES_PFRDMATXSNDSLO[n] (0x00550C00 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMARXSNDSLO	31:0	0x0	RW1C	DYNAMIC	**RDMA Transmitted Sends Low** Counts the number of RDMA Send messages transmitted by the Protocol Engine. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.69 Protocol Engine Statistics RDMA Transmitted Send Messages High - GLPES_PFRDMATXSNDSHI[n] (0x00550C04 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMARXSNDSHI	15:0	0x0	RW1C	DYNAMIC	**RDMA Transmitted Sends High** Counts the number of RDMA Send messages transmitted by the Protocol Engine. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.70 Protocol Engine Statistics RDMA Verbs Bind Operations Low - GLPES_PFRDMAVBNDLO[n] (0x00551400 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMAVBNDLO	31:0	0x0	RW1C	DYNAMIC	**RDMA Verbs Binds Low** Counts a total number of RDMA Verb Bind operations carried out by the Protocol Engine. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.71 Protocol Engine Statistics RDMA Verbs Bind Operations High - GLPES_PFRDMAVBNDHI[n] (0x00551404 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMAVBNDHI	15:0	0x0	RW1C	DYNAMIC	**RDMA Verbs Binds High** Counts a total number of RDMA Verb Bind operations carried out by the Protocol Engine. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.72 Protocol Engine Statistics RDMA Verbs Invalidate Operations Low - GLPES_PFRDMAVINVLO[n] (0x00551C00 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMAVINVLO	31:0	0x0	RW1C	DYNAMIC	**RDMA Verbs Invalidates Low** Counts a total number of RDMA Verb Invalidate operations carried out by the Protocol Engine. This is the low 32 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.73 Protocol Engine Statistics RDMA Verbs Invalidate Operations High - GLPES_PFRDMAVINVHI[n] (0x00551C04 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMAVINVHI	15:0	0x0	RW1C	DYNAMIC	**RDMA Verbs Invalidates High** Counts a total number of RDMA Verb Invalidate operations carried out by the Protocol Engine. This is the high 16 bits of the 48-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.24.74 Protocol Engine Statistics TCP Retransmitted Segments - GLPES_PFTCPRTXSEG[n] (0x00552400 + 0x4*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPRTXSEG	31:0	0x0	RW1C	DYNAMIC	**TCP Retransmitted Segments** Counts the number of TCP segments retransmitted by the Protocol Engine.

13.2.2.24.75 Protocol Engine Statistics Congestion Notification Packets Ignored - GLPES_PFRXRPCNPIGNORED[n] (0x00552800 + 0x4*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RXRPCNPIGNORED	23:0	0x0	RW1C	DYNAMIC	**Congestion Notification Packets Ignored** Counts the number of Congestion Notification Packets (CNPs) that have been ignored by the reaction point.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.24.76 Protocol Engine Statistics Congestion Notification Packets Handled - GLPES_PFRXRPCNPHANDLED[n] (0x00552C00 + 0x4*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RXRPCNPHANDLED	31:0	0x0	RW1C	DYNAMIC	**Congestion Notification Packets Handled** Counts the number of Congestion Notification Packets (CNPs) that have been handled by the reaction point.

13.2.2.24.77 Protocol Engine Statistics with ECN Bits Indicating Congestion Low - GLPES_PFRXNPECNMARKEDPKTSLO[n] (0x00553000 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RXNPECNMARKEDPKTSLO	31:0	0x0	RW1C	DYNAMIC	**Congestion Notification Marked Packets Low** Counts the number of packets that have the ECN bits set to indicate congestion. This is a low 32 bits of the 56-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.78 Protocol Engine Statistics with ECN Bits Indicating Congestion High - GLPES_PFRXNPECNMARKEDPKTSHI[n] (0x00553004 + 0x8*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RXNPECNMARKEDPKTSHI	23:0	0x0	RW1C	DYNAMIC	**Congestion Notification Marked Packets High** Counts the number of packets that have the ECN bits set to indicate congestion. This is a high 24 bits of the 56-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.24.79 Protocol Engine Congestion Indication Sent Count - GLPES_PFTXNPCNPSENT[n] (0x00553800 + 0x4*n, n=0...127; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TXNPCNPSENT	23:0	0x0	RW1C	DYNAMIC	**Congestion Notification Packets Sent** Counts the number of Congestion Notification Packets (CNPs) that have been sent by the reaction point.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.24.80 Protocol Engine Statistics RDMA Received Unaligned FPDUs - GLPES_RDMARXUNALIGN (0x0055E000; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMRXAUNALIGN	31:0	0x0	RW1C	DYNAMIC	**RDMA Received Unaligned** Counts the number of TCP segments received by the Protocol Engine that probably carried unaligned FPDUs.

13.2.2.24.81 Protocol Engine Statistics RDMA Received Out of Order No Markers FPDUs - GLPES_RDMARXOOONOMARK (0x0055E004; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMAOOONOMARK	31:0	0x0	RW1C	DYNAMIC	**RDMA Out-of-Order No Markers** Counts RDMA FPDUs received by the Protocol Engine out-of-order and not carrying markers.

13.2.2.24.82 Protocol Engine Statistics RDMA Received Multiple FPDUs Low - GLPES_RDMARXMULTFPDUSLO (0x0055E008; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMARXMULTFPDUSLO	31:0	0x0	RW1C	DYNAMIC	**RDMA Received Multiple FPDUs Low** Counts the number of TCP segments received by the Protocol Engine that probably have multiple FPDUs. This is a low 32 bits of the 56-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.83 Protocol Engine Statistics RDMA Received Multiple FPDUs High - GLPES_RDMARXMULTFPDUSHI (0x0055E00C; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMARXMULTFPDUSHI	23:0	0x0	RW1C	DYNAMIC	**RDMA Received Multiple FPDUs Low** Counts the number of TCP segments received by the Protocol Engine that probably have multiple FPDUs. This is a high 24 bits of the 56-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.24.84 Protocol Engine Statistics RDMA Out of Order Placed DDP Segments Low - GLPES_RDMARXOOODDPLO (0x0055E010; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMARXOOODDPLO	31:0	0x0	RW1C	DYNAMIC	**RDMA Received Out-of-Order DDP Low** Counts the number of the DDP segments received by the Protocol Engine, and out-of-order placed. This is a low 32 bits of the 56-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.85 Protocol Engine Statistics RDMA Out of Order Placed DDP Segments High - GLPES_RDMARXOOODDPHI (0x0055E014; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMARXOOODDPHI	23:0	0x0	RW1C	DYNAMIC	**RDMA Received Out-of-Order DDP High** Counts the number of the DDP segments received by the Protocol Engine, and out-of-order placed. This is a high 24 bits of the 56-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.24.86 Protocol Engine Statistics TCP Received Pure Acks Low - GLPES_TCPRXPUREACKSLO (0x0055E018; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPRXPUREACKLO	31:0	0x0	RW1C	DYNAMIC	**TCP Received Pure Acks Low** Counts the number of TCP Acks received by the Protocol Engine carrying no data. This is a low 32 bits of the 56-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.87 Protocol Engine Statistics TCP Received Pure Acks High - GLPES_TCPRXPUREACKHI (0x0055E01C; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPRXPUREACKHI	23:0	0x0	RW1C	DYNAMIC	**TCP Received Pure Acks High** Counts the number of TCP Acks received by the Protocol Engine carrying no data. This is a high 24 bits of the 56-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.24.88 Protocol Engine Statistics TCP Receive First Hole Low - GLPES_TCPRXONEHOLELO (0x0055E020; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPRXONEHOLELO	31:0	0x0	RW1C	DYNAMIC	**TCP Receive One Hole Low** Counts the number of TCP segments received by the Protocol Engine and opened a first TCP Hole in TCP sequence space. This is a low 32 bits of the 56-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.89 Protocol Engine Statistics TCP Received First Hole High - GLPES_TCPRXONEHOLEHI (0x0055E024; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPRXONEHOLEHI	23:0	0x0	RW1C	DYNAMIC	**TCP Receive One Hole High** Counts the number of TCP segments received by the Protocol Engine and opened a first TCP Hole in TCP sequence space. This is a high 24 bits of the 56-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.24.90 Protocol Engine Statistics TCP Receive Second Hole Low - GLPES_TCPRXTWOHOLELO (0x0055E028; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPRXTWOHOLELO	31:0	0x0	RW1C	DYNAMIC	**TCP Receive Two Hole Low** Counts the number of TCP segments received by the Protocol Engine and opened a second TCP Hole in TCP sequence space. This is a low 32 bits of the 56-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.91 Protocol Engine Statistics TCP Received Second Hole High - GLPES_TCPRXTWOHOLEHI (0x0055E02C; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPRXTWOHOLEHI	23:0	0x0	RW1C	DYNAMIC	**TCP Receive Two Hole High** Counts the number of TCP segments received by the Protocol Engine and opened a second TCP Hole in TCP sequence space. This is a high 24 bits of the 56-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.24.92 Protocol Engine Statistics TCP Receive Third Hole Low - GLPES_TCPRXTHREEHOLELO (0x0055E030; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPRXTHREEHOLELO	31:0	0x0	RW1C	DYNAMIC	**TCP Receive Three Hole Low** Counts the number of TCP segments received by the Protocol Engine and opened a third TCP Hole in TCP sequence space. This is a low 32 bits of the 56-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.93 Protocol Engine Statistics TCP Received Third Hole High - GLPES_TCPRXTHREEHOLEHI (0x0055E034; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPRXTHREEHOLEHI	23:0	0x0	RW1C	DYNAMIC	**TCP Receive Three Hole High** Counts the number of TCP segments received by the Protocol Engine and opened a third TCP Hole in TCP sequence space. This is a high 24 bits of the 56-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.24.94 Protocol Engine Statistics TCP Receive Fourth Hole Low - GLPES_TCPRXFOURHOLELO (0x0055E038; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPRXFOURHOLELO	31:0	0x0	RW1C	DYNAMIC	**TCP Receive Four Hole Low** Counts the number of TCP segments received by the Protocol Engine and opened a fourth TCP Hole in TCP sequence space. This is a low 32 bits of the 56-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.95 Protocol Engine Statistics TCP Receive Fourth Hole High - GLPES_TCPRXFOURHOLEHI (0x0055E03C; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPRXFOURHOLEHI	23:0	0x0	RW1C	DYNAMIC	**TCP Receive Four Hole High** Counts the number of TCP segments received by the Protocol Engine and opened a fourth TCP Hole in TCP sequence space. This is a high 24 bits of the 56-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.24.96 Protocol Engine Statistics TCP Fast Retransmissions Low - GLPES_TCPTXRETRANSFASTLO (0x0055E040; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPTXRETRANSFASTLO	31:0	0x0	RW1C	DYNAMIC	**TCP Fast Retransmissions Low** Counts the number of TCP Fast Retransmissions by the Protocol Engine. This is a low 32 bits of the 56-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.97 Protocol Engine Statistics TCP Fast Retransmissions High - GLPES_TCPTXRETRANSFASTHI (0x0055E044; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPTXRETRANSFASTHI	23:0	0x0	RW1C	DYNAMIC	**TCP Fast Retransmissions High** Counts the number of TCP Fast Retransmissions by the Protocol Engine. This is a high 24 bits of the 56-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.24.98 Protocol Engine Statistics TCP Fast Retransmission Timeouts Low - GLPES_TCPTXTOUTSFASTLO (0x0055E048; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPTXTOUTSFASTLO	31:0	0x0	RW1C	DYNAMIC	**TCP Fast Retransmission Timeouts Low** Counts the number of TCP retransmission timeouts by the Protocol Engine on the connections attempting fast retransmit. This is a low 32 bits of the 56-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.99 Protocol Engine Statistics TCP Fast Retransmissions Timeouts High - GLPES_TCPTXTOUTSFASTHI (0x0055E04C; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPTXTOUTSFASTHI	23:0	0x0	RW1C	DYNAMIC	**TCP Fast Retransmission Timeouts High** Counts the number of TCP retransmission timeouts by the Protocol Engine on the connections attempting fast retransmit. This is a high 24 bits of the 56-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.24.100 Protocol Engine Statistics TCP Retransmission Timeouts Low - GLPES_TCPTXTOUTSLO (0x0055E050; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPTXTOUTSLO	31:0	0x0	RW1C	DYNAMIC	**TCP Retransmission Timeouts Low** Counts the number of TCP retransmission timeouts by the Protocol Engine on the connections that are not currently attempting retransmit. This is a low 32 bits of the 56-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.

13.2.2.24.101 Protocol Engine Statistics TCP Retransmissions Timeouts High - GLPES_TCPTXTOUTSHI (0x0055E054; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPTXTOUTSHI	23:0	0x0	RW1C	DYNAMIC	**TCP Retransmission Timeouts High** Counts the number of TCP retransmission timeouts by the Protocol Engine on the connections that are not currently attempting retransmit. This is a high 24 bits of the 56-bit counter. The low and high registers are part of a 64-bit register and are read using 64-bit read accesses only. It is implemented internally breaking the read request into two 32-bit reads. Reading the low 32 bits latches the high 32 bits into a shadow register. Reading the high 32 bits returns the value in the shadow register.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.25 PF - Comm Transmit Queues Registers

13.2.2.25.1 Global Transmit Queue Head - QTX_COMM_HEAD[DBQM] (0x000E0000 + 0x4*DBQM, DBQM=0...16383; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	31:0	0x0	RSV	N/A	Reserved.

13.2.2.25.2 Transmit Comm Scheduler Completion Queue Control - GLCOMM_CQ_CTL[CQ] (0x000F0000 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
COMP_TYPE	2:0	000b	SC	UNDEFINED	**Completion Type** *COMP_TYPE* reported on marker when command opcode = 1.
RESERVED	3	0b	RSV	N/A	Reserved.
CMD	6:4	000b	SC	UNDEFINED	**Command** Command Opcode as follows: 000b = VM Reset marker operation. Initiating this command, VM reset completion (Completion Type = 6) is written with the VM ID provided in the *ID* field. The completion queue cache line is written to the host immediately (like in ITR event). 001b = Write current completion cache line to the host and send marker with "Completion Type" taken from *COMP_TYPE* field and VM/Q ID taken from the *ID* field. The completion queue cache line is written to the host immediately (like in ITR event). All other values are reserved.
RESERVED	15:7	0x0	RSV	N/A	Reserved.
ID	29:16	0x0	SC	UNDEFINED	**ID** When Command Opcode == 0, this field represents the VM ID and is copied to VM/Q ID field in CQ descriptor. When Command Opcode == 1, this field is copied to VM/Q ID field in CQ descriptor.
RESERVED	31:30	00b	RSV	N/A	Reserved.

13.2.2.25.3 Tx Completion Queue Context Register 0 - GLTCLAN_CQ_CNTX0[CQ] (0x000F0800 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RING_ADDR_LSB	31:0	0x0	RW	UNDEFINED	**Ring Address LSB** 32 LS bits of the completion queue base address. The resolution is 128 bytes.

13.2.2.25.4 Tx Completion Queue Context Register 1 - GLTCLAN_CQ_CNTX1[CQ] (0x000F1000 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RING_ADDR_MSB	24:0	0x0	RW	UNDEFINED	**Ring Address MSB** 25 MS bits of the completion queue base address. The resolution is 128 bytes.
RESERVED	31:25	0x0	RSV	N/A	Reserved.

13.2.2.25.5 Tx Completion Queue Context Register 2 - GLTCLAN_CQ_CNTX2[CQ] (0x000F1800 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RING_LEN	17:0	0x0	RW	UNDEFINED	**Ring Length** Completion queue length in 16 completions unit (64 bytes).
RESERVED	31:18	0x0	RSV	N/A	Reserved.

13.2.2.25.6 Tx Completion Queue Context Register 3 - GLTCLAN_CQ_CNTX3[CQ] (0x000F2000 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GENERATION	0	0b	RW	UNDEFINED	**Generation** Tx completion queue generation bit exposed on CQ write-back. Value flips each ring wrap-around.
CQ_WR_PTR	22:1	0x0	RW	UNDEFINED	**CQ Write Pointer**
RESERVED	31:23	0x0	RSV	N/A	Reserved.

13.2.2.25.7 Tx Completion Queue Context Register 4 - GLTCLAN_CQ_CNTX4[CQ] (0x000F2800 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PF_NUM	2:0	000b	RW	UNDEFINED	**PF Number** PF number of the CQ.
VMVF_NUM	12:3	0x0	RW	UNDEFINED	**VM/VF Number** VM number or VF number.
VMVF_TYPE	14:13	00b	RW	UNDEFINED	**VM/VF Type** 00b = CQ belongs to a VF. *VMVF_NUM* describes the VM number. 01b = CQ belongs to a VM. *VMVF_NUM* describes the VM number. 10b = CQ is a PF only CQ. *VMVF_NUM* is meaningless. 11b = Reserved.
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.25.8 Tx Completion Queue Context Register 5 - GLTCLAN_CQ_CNTX5[CQ] (0x000F3000 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TPH_EN	0	0b	RW	UNDEFINED	**TPH Enable** TPH enable for CQ write accesses.
CPU_ID	8:1	0x0	RW	UNDEFINED	**CPU ID** CPUID for CQ write accesses.
FLUSH_ON_ITR_DIS	9	0b	RW	UNDEFINED	**Flush on ITR Discard** 1b = CQ flush when ITR expire is silently dropped.
RESERVED	31:10	0x0	RSV	N/A	Reserved.

13.2.2.25.9 Tx Completion Queue Context Register 6 - GLTCLAN_CQ_CNTX6[CQ] (0x000F3800 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CQ_CACHLINE	31:0	0x0	RW	UNDEFINED	**CQ Cache Line** 32-bit cache line raw data.

13.2.2.25.10 Tx Completion Queue Context Register 7 - GLTCLAN_CQ_CNTX7[CQ] (0x000F4000 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CQ_CACHLINE	31:0	0x0	RW	UNDEFINED	**CQ Cache Line** 32-bit cache line raw data.

13.2.2.25.11 Tx Completion Queue Context Register 8 - GLTCLAN_CQ_CNTX8[CQ] (0x000F4800 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CQ_CACHLINE	31:0	0x0	RW	UNDEFINED	**CQ Cache Line** 32-bit cache line raw data.

13.2.2.25.12 Tx Completion Queue Context Register 9 - GLTCLAN_CQ_CNTX9[CQ] (0x000F5000 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CQ_CACHLINE	31:0	0x0	RW	UNDEFINED	**CQ Cache Line** 32-bit cache line raw data.

13.2.2.25.13 Tx Completion Queue Context Register 10 - GLTCLAN_CQ_CNTX10[CQ] (0x000F5800 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CQ_CACHLINE	31:0	0x0	RW	UNDEFINED	**CQ Cache Line** 32-bit cache line raw data.

13.2.2.25.14 Tx Completion Queue Context Register 11 - GLTCLAN_CQ_CNTX11[CQ] (0x000F6000 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CQ_CACHLINE	31:0	0x0	RW	UNDEFINED	**CQ Cache Line** 32-bit cache line raw data.

13.2.2.25.15 Tx Completion Queue Context Register 12 - GLTCLAN_CQ_CNTX12[CQ] (0x000F6800 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CQ_CACHLINE	31:0	0x0	RW	UNDEFINED	**CQ Cache Line** 32-bit cache line raw data.

13.2.2.25.16 Tx Completion Queue Context Register 13 - GLTCLAN_CQ_CNTX13[CQ] (0x000F7000 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CQ_CACHLINE	31:0	0x0	RW	UNDEFINED	**CQ Cache Line** 32-bit cache line raw data.

13.2.2.25.17 Tx Completion Queue Context Register 14 - GLTCLAN_CQ_CNTX14[CQ] (0x000F7800 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CQ_CACHLINE	31:0	0x0	RW	UNDEFINED	**CQ Cache Line** 32-bit cache line raw data.

13.2.2.25.18 Tx Completion Queue Context Register 15 - GLTCLAN_CQ_CNTX15[CQ] (0x000F8000 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CQ_CACHLINE	31:0	0x0	RW	UNDEFINED	**CQ Cache Line** 32-bit cache line raw data.

13.2.2.25.19 Tx Completion Queue Context Register 16 - GLTCLAN_CQ_CNTX16[CQ] (0x000F8800 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CQ_CACHLINE	31:0	0x0	RW	UNDEFINED	**CQ Cache Line** 32-bit cache line raw data.

13.2.2.25.20 Tx Completion Queue Context Register 17 - GLTCLAN_CQ_CNTX17[CQ] (0x000F9000 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CQ_CACHLINE	31:0	0x0	RW	UNDEFINED	**CQ Cache Line** 32-bit cache line raw data.

13.2.2.25.21 Tx Completion Queue Context Register 18 - GLTCLAN_CQ_CNTX18[CQ] (0x000F9800 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CQ_CACHLINE	31:0	0x0	RW	UNDEFINED	**CQ Cache Line** 32-bit cache line raw data.

13.2.2.25.22 Tx Completion Queue Context Register 19 - GLTCLAN_CQ_CNTX19[CQ] (0x000FA000 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CQ_CACHLINE	31:0	0x0	RW	UNDEFINED	**CQ Cache Line** 32-bit cache line raw data.

13.2.2.25.23 Tx Completion Queue Context Register 20 - GLTCLAN_CQ_CNTX20[CQ] (0x000FA800 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CQ_CACHLINE	31:0	0x0	RW	UNDEFINED	**CQ Cache Line** 32-bit cache line raw data.

13.2.2.25.24 Tx Completion Queue Context Register 21 - GLTCLAN_CQ_CNTX21[CQ] (0x000FB000 + 0x4*CQ, CQ=0...511; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CQ_CACHLINE	31:0	0x0	RW	UNDEFINED	**CQ Cache Line** 32-bit cache line raw data.

13.2.2.25.25 Global Transmit Comm Min/Max Packet - GLCOMM_MIN_MAX_PKT (0x000FC064; RW)

This register is not expected to be accessed by the software.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MAHDL	13:0	0x2600	RW	UNDEFINED	**Maximum Host Data Length** Maximum transmit packet size in Host buffers (defined in byte units).
RESERVED	15:14	00b	RSV	N/A	Reserved.
MIHDL	21:16	0x11	RW	UNDEFINED	**Minimum Host Data Length** Minimum transmit packet size in Host buffers (defined in byte units).
LSO_COMS_MIHDL	31:22	0x0	RW	UNDEFINED	Reserved.

13.2.2.25.26 Transmit Comm Scheduler Tx LAN Cache Control - GLLAN_TCLAN_CACHE_CTL (0x000FC0B8; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MIN_FETCH_THRESH	5:0	0x8	RW	UNDEFINED	**Minimum Fetch Threshold** Minimum fetch threshold in LAN mode. Descriptor fetch length is rounded up to this value.
RESERVED	6	1b	RSV	N/A	Reserved.
MIN_ALLOC_THRESH	13:7	0x18	RW	UNDEFINED	**Minimum Allocated Threshold** Minimum number of allocated descriptors in the cache for each cache entry. Applicable for both LAN and Comms mode.
RESERVED	21:14	0x0	RSV	N/A	Reserved.
RESERVED	31:22	0x240	RSV	N/A	Reserved.

13.2.2.25.27 Transmit Comm Scheduler Queue Doorbell - QTX_COMM_DBELL[DBQM] (0x002C0000 + 0x4*DBQM, DBQM=0...16383; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
QTX_COMM_DBELL	31:0	0x0	RW	UNDEFINED	**QTX Comms Doorbell** See Section 10.5 for complete description of the SSO/LSO/DROP doorbell formats. ***Note:*** This CSR always returns 0x0 when read and not the written value. To get the *TAIL* value of some Tx-Queue, read it from Tx-Queue context.

13.2.2.25.28 Transmit Comm Scheduler Queue Context - QTX_COMM_DBLQ_CNTX[n,DBLQ] (0x002D0000 + 0x400*n + 0x4*DBLQ, n=0...4, DBLQ=0...255; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DATA	31:0	0x0	RW	UNDEFINED	**Data** Comm DBL Queue context data. Bit 'i' in register 'n' is mapped to bit “32*n+i” in the comm DBL queue context described in Section 10.5.5.6.

13.2.2.25.29 Transmit Comm Scheduler Queue Doorbell - QTX_COMM_DBLQ_DBELL[DBLQ] (0x002D1400 + 0x4*DBLQ, DBLQ=0...255; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TAIL	12:0	0x0	RW	UNDEFINED	**Tail** Defines the first DBLQ DBL that the software prepares for the hardware (it is the last valid DBLQ DBL plus one). The DBLQ Tail is a relative descriptor index to the beginning of the DBLQ ring. This field is applicable for Comms mode only.
RESERVED	31:13	0x0	RSV	N/A	Reserved.

13.2.2.25.30 Transmit Comm Scheduler Queue Context Data - GLCOMM_QTX_CNTX_DATA[n] (0x002D2D40 + 0x4*n, n=0...9; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DATA	31:0	0x0	RW	UNDEFINED	**Data** Comm Transmit Queue context data. Bit 'i' in register 'n' is mapped to bit “32*n+i” in the comm transmit queue context described in Section 10.5.5.2.1.

13.2.2.25.31 Global Transmit Comm Scheduler Quanta Profile - GLCOMM_QUANTA_PROF[n] (0x002D2D68 + 0x4*n, n=0...15; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
QUANTA_SIZE	13:0	0x0	RW	UNDEFINED	**Quanta Size** Configured Quanta size in bytes. Used by hardware for LSO processing and for legacy Queue. When it configures a legacy Queue Quanta size, the value must be configured in 64-byte granularity (The five LS bits must be set to 0).
RESERVED	15:14	00b	RSV	N/A	Reserved.
MAX_CMD	23:16	0x0	RW	UNDEFINED	**Maximum Commands** Max commands generated in a single Quanta. When a specific quanta exceeds this number, the Tx-Queue is disabled and the PF is notified by CSR GL_MDET_TX_TCLAN.
MAX_DESC	29:24	0x0	RW	UNDEFINED	**Maximum Descriptors** Max descriptors in a single Quanta. When a specific quanta exceeds this number, the Tx-Queue is disabled and the PF is notified by CSR GL_MDET_TX_TCLAN.
RESERVED	31:30	00b	RSV	N/A	Reserved.

13.2.2.25.32 Global Transmit Comm Scheduler Quanta Profile - GLCOMM_PKT_SHAPER_PROF[n] (0x002D2DA8 + 0x4*n, n=0...7; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PKTCNT	5:0	0x0	RW	UNDEFINED	**Packet Count** Configured Quanta size in packets. Used in Legacy Host interface Queue.
RESERVED	31:6	0x0	RSV	N/A	Reserved.

13.2.2.25.33 Transmit Comm Scheduler Queue Context Control - GLCOMM_QTX_CNTX_CTL (0x002D2DC8; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
QUEUE_ID	13:0	0x0	RW	UNDEFINED	**Queue ID** Transmit queue ID accessed by the GLCSCD_TXQ_CNTX registers.
RESERVED	15:14	00b	RSV	N/A	Reserved.
CMD	18:16	000b	RW	UNDEFINED	**Command** Command Opcode as follows: 000b = Read operation. 001b = Write operation, update static and dynamic fields. 011b = Reset Transmit queue context state, updates all static fields. 100b = Update static fields, preserving dynamic fields. All other values are reserved.
CMD_EXEC	19	0b	RW	UNDEFINED	**Command Execute** Setting the *CMD_EXEC* flag kicks off the operation.
RESERVED	31:20	0x0	RSV	N/A	Reserved.

13.2.2.25.34 Transmit Comm Scheduler Queue Context Status - GLCOMM_QTX_CNTX_STAT (0x002D2DCC; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CMD_IN_PROG	0	0b	RO	N/A	**Command in Progress** Initiating a command to the GLCSCD_TXQ_CTL register (setting the *CMD_EXEC* flag), the device sets the *CMD_IN_PROG* flag. Then, the *CMD_IN_PROG* flag is cleared when the command is completed.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.26 PF - LAN Transmit/Receive Registers

13.2.2.26.1 L2 Tag - Enable - PRT_TDPUL2TAGSEN (0x00040BA0; RW)

L2 tags functionality enable per port.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ENABLE	7:0	0x0	RW	UNDEFINED	**Enable** Defines the L2 tags expected on this port.
NONLAST_TAG	15:8	0x8	RW	UNDEFINED	**Non-Last Tags** For each L2 tag, if tag with same eth_type is expected following this tag, set the corresponding bit in this field to "1". Otherwise, set it to zero. Default is 0x04 to support double VLAN. ***Note:*** This field is implemented only in the TDPU version of this register.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.26.2 Transmit DCSP to TC Enforcement - IPv4 - GL_HLP_PRT_IPG_PREAMBLE_SIZE[n] (0x00049240 + 0x4*n, n=0...20; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
IPG_PREAMBLE_SIZE	7:0	0x14	RW	UNDEFINED	**IPG+Preamble Size** IPG+Preamble size for credit update adjustments to scheduler. Configurable per HLP port.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.26.3 Transmit TDPU Scheduler 4 Adjustment Default Recipe - GL_TDPU_PSM_DEFAULT_RECIPE[n] (0x00049294 + 0x4*n, n=0...3; RO)

TDPU default recipe to scheduler credits four adjustments (default recipe and mode per adjustment).

Field	Bit(s)	Init.	Type	CFG Policy	Description
ADD_IPG	0	0b	RW	UNDEFINED	**Add IPG** 1b = The default recipe includes IPG+Preamble (as configured in HLP_PRT_IPG_PREAMBLE size).
SUB_CRC	1	1b	RW	UNDEFINED	**Subtract CRC** 0b = The default recipe subtracts four MAC CRC bytes.
SUB_ESP_TRAILER	2	1b	RW	UNDEFINED	**Subtract ESP Trailer** 0b = The default recipe subtracts ESP trailer length.
INCLUDE_L2_PAD	3	1b	RW	UNDEFINED	**Include L2 Padding** 0b = The default recipe subtracts the number of L2 padding bytes that were in the packet received by the host. 1b = The default recipe adds L2 padding added by the Transmit data processing unit.

Field	Bit(s)	Init.	Type	CFG Policy	Description

DEFAULT_UPDATE_MODE	4	0b	RW	UNDEFINED	**Default Update Mode**
					Default update mode (upon default recipe).
					0b = For update #N, if no recipe is provided, give same update as for update #N-1. For update #0, use default recipe (also add L2 tags and inner VLAN if were inserted by Tx).
					1b = For update #N, if no recipe is provided, use default recipe as configured in this CSR. (also add L2 tags and inner VLAN if were inserted by Tx).

RESERVED	31:5	0x0	RSV	N/A	Reserved.

13.2.2.26.4 Global TSO TCP Mask First - GLLAN_TSOMSK_F (0x00049308; RO)

TSO TCP first-segment Flags mask control.

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPMSKF	11:0	0x9	RW	UNDEFINED	**TCP Mask First** TCP Flags mask for the first segment in the TSO. Any bit set to one in the *TCPMSKF* field clears the respective TCP flag in the TSO. Bit zero relates to 'FIN' flag, bit one relates to the 'SYN' flag, and so on.
RESERVED	31:12	0x0	RSV	N/A	Reserved.

13.2.2.26.5 Global TSO TCP Mask Middle - GLLAN_TSOMSK_M (0x0004930C; RO)

TSO TCP middle-segments Flags mask control.

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPMSKM	11:0	0x89	RW	UNDEFINED	**TCP Mask Middle** TCP Flags mask for the middle segments in the TSO. See *TCPMSKF* for the impact of each bit in the field.
RESERVED	31:12	0x0	RSV	N/A	Reserved.

13.2.2.26.6 Global TSO TCP Mask Last - GLLAN_TSOMSK_L (0x00049310; RO)

TSO TCP last-segment Flags mask control.

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCPMSKL	11:0	0x80	RW	UNDEFINED	**TCP Mask Last** TCP Flags mask for the last segment in the TSO. See *TCPMSKF* for the impact of each bit in the field.
RESERVED	31:12	0x0	RSV	N/A	Reserved.

13.2.2.26.7 VF PF Rx-Queue Mapping Table - VPLAN_RX_QTABLE[n,VF] (0x00060000 + 0x800*n + 0x4*VF, n=0...15, VF=0...255; RW)

This register affects the VF but exposed only to the parent PF.

Field	Bit(s)	Init.	Type	CFG Policy	Description
QINDEX	11:0	0xFFF	RW	UNDEFINED	**Queue Index** Defines the index of VF queue 'n' in the PF queues space, where 'n' is the register index. Setting the *QINDEX* to 0xFFF means that the queue is not valid for the VF. Relevant only if VPLAN_QBASE.*VFQTABLE_ENA* is set.
RESERVED	31:12	0x0	RSV	N/A	Reserved.

13.2.2.26.8 VF PF DB Queue Mapping Table - VPLAN_DB_QTABLE[n,VF] (0x00070000 + 0x800*n + 0x4*VF, n=0...3, VF=0...255; RW)

This register affects the VF but exposed only to the parent PF.

Field	Bit(s)	Init.	Type	CFG Policy	Description
QINDEX	8:0	0x1FF	RW	UNDEFINED	**Queue Index** Defines the index of VF queue 'n' in the PF queues space, where 'n' is the register index. Setting the QINDEX to 0x1FF means that the queue is not valid for the VF. Relevant only if VPLAN_QBASE.*VFQTABLE_ENA* is set.
RESERVED	31:9	0x0	RSV	N/A	Reserved.

13.2.2.26.9 VF PF Rx-Queue Range - VPLAN_RX_QBASE[VF] (0x00072000 + 0x4*VF, VF=0...255; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VFFIRSTQ	10:0	0x0	RW	UNDEFINED	**VF First Queue** Defines the base index of VF 'n' within the range of the PF queues, where 'n' is the VF register index. The *VFFIRSTQ* field is meaningful for this VF only if *VFQTABLE_ENA* is cleared.
RESERVED	15:11	0x0	RSV	N/A	Reserved.
VFNUMQ	23:16	0x0	RW	UNDEFINED	**VF Number of Queues** Defines the number of queues allocated to VF 'n' within the range of the PF queues, where 'n' is the VF register index. The *VFNUMQ* is meaningful for this VF only if *VFQTABLE_ENA* is cleared. The value should be set to the number of queues. A value of 0 means 1 queue and a value of 255 means 256 queues.
RESERVED	30:24	0x0	RSV	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
VFQTABLE_ENA	31	1b	RW	UNDEFINED	**VSI Queue Table Enable** Selects between contiguous range of queues for this VSI vs. scattered range: 0b = The VSI is assigned a contiguous range starting at *VSIBASE*. 1b = The VSI is assigned a scattered range defined by the VSILAN_QTABLE. ***Note:*** The default of this bit is 1b to keep backward compatibility with the X710/XXV710/XL710.

13.2.2.26.10 VF LAN RXQ Enablement - VPLAN_RXQ_MAPENA[VF] (0x00073000 + 0x4*VF, VF=0...255; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RX_ENA	0	0b	RW	UNDEFINED	**Rx Enable** The VPLAN_RX_QTABLE or VPLAN_RX_QBASE for the VF are enabled only when the *RX_ENA* flag is set.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.26.11 VF LAN TXQ Enablement - VPLAN_TXQ_MAPENA[VF] (0x00073800 + 0x4*VF, VF=0...255; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TX_ENA	0	0b	RW	UNDEFINED	**Tx Enable** The VPLAN_TX_QTABLE or VPLAN_TX_QBASE for the VF are enabled only when the *TX_ENA* flag is set.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.26.12 VF PF Rx-Queue Mapping Table - VPDSI_RX_QTABLE[n,VP16] (0x00074C00 + 0x40*n + 0x4*VP16, n=0...15, VP16=0...15; RW)

This register affects the VF but exposed only to the parent PF.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PAGE_INDEX0	6:0	0x7F	RW	UNDEFINED	**Page Index 0** Defines the index of a 256-Queues Page that is associated with the queue (@VF Address Space). The value 0x7F stands for invalid 32-Queues Page.
RESERVED	7	0b	RSV	N/A	Reserved.
PAGE_INDEX1	14:8	0x7F	RW	UNDEFINED	**Page Index 1** Defines the index of a 32-Queues Page that is associated with the queue (@VF Address Space). The value 0x7F stands for invalid 256-Queues Page.
RESERVED	15	0b	RSV	N/A	Reserved.
PAGE_INDEX2	22:16	0x7F	RW	UNDEFINED	**Page Index 2** Defines the index of a 32-Queues Page that is associated with the queue (@VF Address Space). The value 0x7F stands for invalid 256-Queues Page.
RESERVED	23	0b	RSV	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PAGE_INDEX3	30:24	0x7F	RW	UNDEFINED	**Page Index 3** Defines the index of a 32-Queues Page that is associated with the queue (@VF Address Space). The value 0x7F stands for invalid 256-Queues Page.
RESERVED	31	0b	RSV	N/A	Reserved.

13.2.2.26.13 PF Doorbell Queue Allocation - PFLAN_DB_QALLOC (0x00075680; RW)

These registers define the LAN queue pairs allocation to the PFs.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FIRSTQ	7:0	0x0	RW	UNDEFINED	**First Queue** The first Tx LAN Queue pair allocated to this PF. Valid only if the *VALID* flag is set. Valid values are 0-16383.
RESERVED	15:8	0x0	RSV	N/A	Reserved.
LASTQ	23:16	0x0	RW	UNDEFINED	**Last Queue** The last Tx LAN Queue pair allocated to this PF. Valid only if the *VALID* flag is set. Valid values are 0-16383.
RESERVED	30:24	0x0	RSV	N/A	Reserved.
VALID	31	0b	RW	UNDEFINED	**VALID** Indicates that queues are allocated to this PF. For any active PF, this flag must be set.

13.2.2.26.14 PF Completion Queue Allocation - PFLAN_CP_QALLOC (0x00075700; RW)

These registers define the LAN queue pairs allocation to the PFs.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FIRSTQ	8:0	0x0	RW	UNDEFINED	**First Queue** The first Tx LAN Queue pair allocated to this PF. Valid only if the *VALID* flag is set. Valid values are 0-16383.
RESERVED	15:9	0x0	RSV	N/A	Reserved.
LASTQ	24:16	0x0	RW	UNDEFINED	**Last Queue** The last Tx LAN Queue pair allocated to this PF. Valid only if the *VALID* flag is set. Valid values are 0-16383.
RESERVED	30:25	0x0	RSV	N/A	Reserved.
VALID	31	0b	RW	UNDEFINED	**VALID** Indicates that queues are allocated to this PF. For any active PF, this flag must be set.

13.2.2.26.15 Global Receive Queue Control - QRX_CTRL[QRX] (0x00120000 + 0x4*QRX, QRX=0...2047; RW)

The access type of specific fields in this register are determined individually. It is documented per field in the internal registers space.

Field	Bit(s)	Init.	Type	CFG Policy	Description
QENA_REQ	0	0b	RW	UNDEFINED	**Queue Enable Request** Receive queue enable request. Setting this bit the software should poll the *QENA_STAT* flag (in this register) before using the queue. After clearing this flag the software should poll the *QENA_STAT* flag before releasing the memory structures. Once the software changes the state of the *QENA_REQ* flag, it must poll the *QENA_STAT* flag before it is permitted to revert the state of the *QENA_REQ* once again.
FAST_QDIS	1	0b	RW1C	DYNAMIC	**Fast Queue Disable** See Section 10.4.3.1.2.1 for the usage of this flag. This flag is auto-cleared by the hardware.
QENA_STAT	2	0b	RO	N/A	**Queue Enable Status** Receive queue enable status indication. 0b = Indicated that the queue is inactive. 1b = Indicates that the queue is active.
CDE	3	0b	RW	UNDEFINED	**Conditional Drop Enable** When set, temporarily enables drop of packets of this queue passing through a no droppable TC. The dropping starts with "no drop" TC experiencing TO event for packet of this queue. This can be reversed by software/firmware by resetting the *CDS* bit.
CDS	4	0b	RW1C	DYNAMIC	**Conditional Drop Status** When set, packets of this queue passing through a no droppable TC can be dropped if required. Set event - "no drop" TC experiencing TO event for packet of this queue. Clear event - software writes 1 to this bit.
RESERVED	31:5	0x0	RSV	N/A	Reserved.

13.2.2.26.16 VF PF Tx-Queue Mapping Table - VPLAN_TX_QTABLE[n,VF] (0x001C0000 + 0x800*n + 0x4*VF, n=0...15, VF=0...255; RW)

This register affects the VF but exposed only to the parent PF.

Field	Bit(s)	Init.	Type	CFG Policy	Description
QINDEX	14:0	0x7FFF	RW	UNDEFINED	**Queue Index** Defines the index of VF queue 'n' in the PF queues space, where 'n' is the register index. Setting the *QINDEX* to 0x7FFF means that the queue is not valid for the VF. Relevant only if VPLAN_QBASE.*VFQTABLE_ENA* is set.
RESERVED	31:15	0x0	RSV	N/A	Reserved.

13.2.2.26.17 VF PF Tx-Queue Range - VPLAN_TX_QBASE[VF] (0x001D1800 + 0x4*VF, VF=0...255; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VFFIRSTQ	13:0	0x0	RW	UNDEFINED	**VF First Queue** Defines the base index of VF 'n' within the range of the PF queues, where 'n' is the VF register index. The VFFIRSTQ field is meaningful for this VSI only if the *VFQTABLE_ENA* flag is cleared.
RESERVED	15:14	00b	RSV	N/A	Reserved.
VFNUMQ	23:16	0x0	RW	UNDEFINED	**VF Number of Queues** Defines the number of queue allocated to VF 'n' within the range of the PF queues, where 'n' is the VF register index. The VFNUMQ field is meaningful for this VSI only if the *VFQTABLE_ENA* flag is cleared. The value should be set to the number of queues -1 (A value of 0 means 1 queue and a value of 255 means 256 queues).
RESERVED	30:24	0x0	RSV	N/A	Reserved.
VFQTABLE_ENA	31	1b	RW	UNDEFINED	**VF Queue Table Enable** Selects between contiguous range of queues for this VSI vs. scattered range: 0b = The VF is assigned a contiguous range starting at *VFFIRSTQ*. 1b = The VF is assigned a scattered range defined by the VFLAN_QTABLE. ***Note:*** The default of this bit is 1b to keep backward compatibility with the X710/XXV710/XL710.

13.2.2.26.18 VF PF Tx-Queue Mapping Table - VPDSI_TX_QTABLE[n,VP16] (0x001D2000 + 0x40*n + 0x4*VP16, n=0...15, VP16=0...15; RW)

This table maps blocks of queues to DSI VFs.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PAGE_INDEX0	6:0	0x7F	RW	UNDEFINED	**Page Index 0** Defines the index of a 256-Queues Page that is associated with the queue (@VF Address Space). The value 0x7F stands for invalid 256-Queues Page.
RESERVED	7	0b	RSV	N/A	Reserved.
PAGE_INDEX1	14:8	0x7F	RW	UNDEFINED	**Page Index 1** Defines the index of a 256-Queues Page that is associated with the queue (@VF Address Space). The value 0x7F stands for invalid 256-Queues Page.
RESERVED	15	0b	RSV	N/A	Reserved.
PAGE_INDEX2	22:16	0x7F	RW	UNDEFINED	**Page Index 2** Defines the index of a 256-Queues Page that is associated with the queue (@VF Address Space). The value 0x7F stands for invalid 256-Queues Page.
RESERVED	23	0b	RSV	N/A	Reserved
PAGE_INDEX3	30:24	0x7F	RW	UNDEFINED	**Page Index 3** Defines the index of a 256-Queues Page that is associated with the queue (@VF Address Space). The value 0x7F stands for invalid 256-Queues Page.
RESERVED	31	0b	RSV	N/A	Reserved.

13.2.2.26.19 PF Queue Allocation - PFLAN_RX_QALLOC (0x001D2500; RW)

These registers define the LAN Rx-Queues allocation to the PFs.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FIRSTQ	10:0	0x0	RW	UNDEFINED	**First Queue** The first LAN Queue pair allocated to this PF. Valid only if the *VALID* flag is set. Valid values are 0-2047.
RESERVED	15:11	0x0	RSV	N/A	Reserved.
LASTQ	26:16	0x0	RW	UNDEFINED	**Last Queue** The last LAN Queue pair allocated to this PF. Valid only if the *VALID* flag is set. Valid values are 0-2047.
RESERVED	30:27	0x0	RSV	N/A	Reserved.
VALID	31	0b	RW	UNDEFINED	**Valid** Indicates that queues are allocated to this PF. For any active PF, this flag must be set.

13.2.2.26.20 PF Queue Allocation - PFLAN_TX_QALLOC (0x001D2580; RW)

These registers define the LAN Tx-Queues allocation to the PFs.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FIRSTQ	13:0	0x0	RW	UNDEFINED	**First Queue** The first Tx LAN Queue pair allocated to this PF. Valid only if the *VALID* flag is set. Valid values are 0-16383.
RESERVED	15:14	00b	RSV	N/A	Reserved.
LASTQ	29:16	0x0	RW	UNDEFINED	**Last Queue** The last Tx LAN Queue pair allocated to this PF. Valid only if the *VALID* flag is set. Valid values are 0-16383.
RESERVED	30	0b	RSV	N/A	Reserved.
VALID	31	0b	RW	UNDEFINED	**Valid** Indicates that queues are allocated to this PF. For any active PF, this flag must be set.

13.2.2.26.21 Global Receive Queue Context - QRX_CONTEXT[n,QRX] (0x00280000 + 0x2000*n + 0x4*QRX, n=0...7, QRX=0...2047; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RXQ_CONTEXT	31:0	0x0	RW	UNDEFINED	**Receive Queues Context** Composed of eight registers per queue. Bit 'i' (i = 0...31) of register 'n' (n=0...7) defines bit "i + 32*n" in the receive queue context as defined in Section 10.4.3.6 and its subsections. Context structure definition includes several reserved fields. These fields must be set to zeros.

13.2.2.26.22 Receive Queue Tail Update - QRX_TAIL[QRX] (0x00290000 + 0x4*QRX, QRX=0...2047; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TAIL	12:0	0x0	RW	UNDEFINED	**Tail** Defines the first descriptor that the software hands to the hardware (it is the last valid descriptor plus one). The Tail is a relative descriptor index to the beginning of the receive descriptor ring.
RESERVED	31:13	0x0	RSV	N/A	Reserved.

13.2.2.26.23 Global Receive Queue ITR Expire - QRX_ITR[QRX] (0x00292000 + 0x4*QRX, QRX=0...2047; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
NO_EXPR	0	0b	RW	UNDEFINED	**No Expire** When set, each completed descriptor is written immediately to host memory.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.26.24 Global RLAN Control 0 - GLLAN_RCTL_0 (0x002941F8; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PXE_MODE	0	1b	RW1C	DYNAMIC	**PXE Mode** When the *PXE_MODE* flag is set, the device fetches and writes back a single descriptor at a time. During nominal performance operation, (non-PXE mode), this flag must be cleared.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.26.25 Global RLAN Control 1 - GLLAN_RCTL_1 (0x002941FC; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	11:0	0x0	RSV	N/A	Reserved.
RXMAX_EXPANSION	15:12	0x0	RW	UNDEFINED	**Rx Maximum Expansion** Maximum number of bytes that can be added to a single receive frame by the hardware offload engines. The field is defined in 32-byte granularity. This field must be set to 0.
RESERVED	16	0b	RSV	UNDEFINED	Reserved.
RXDRDCTL	17	0b	RW	UNDEFINED	**Receive Descriptor Read Control** 0b = Read only those descriptors starting after the latest completed one till the end of the cache line. 1b = Always read whole cache lines.
RXDESCRDROEN	18	0b	RW	UNDEFINED	**Rx-Descriptor Read Relaxed Order Enable** Enables relaxed ordering for Rx-Descriptor reads. 0b = Relaxed ordering is disabled for Rx-Descriptor reads. 1b = Relaxed ordering is enabled for Rx-Descriptor reads.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RXDATAWRROEN	19	0b	RW	UNDEFINED	**Rx Data Write Relaxed Order Enable** Enables relaxed ordering for Rx data writes. 0b = Relaxed ordering is disabled for Rx data writes. 1b = Relaxed ordering is enabled for Rx data writes.
RESERVED	31:20	0x0	RSV	N/A	Reserved.

13.2.2.26.26 Global PF LAN Recipe - GLLAN_PF_RECIPE[n] (0x0029420C + 0x4*n, n=0...7; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RECIPE	1:0	00b	RW	UNDEFINED	**Recipe** Recipe field per PF. Register 'n' is associated with PF 'n'.
RESERVED	31:2	0x0	RSV	N/A	Reserved.

13.2.2.26.27 VF LAN TXQ Enablement - VPLAN_DSI_VF_MODE[VP16] (0x002D2C00 + 0x4*VP16, VP16=0...15; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
LAN_DSI_VF_MODE	0	0b	RW	UNDEFINED	**LAN DSI VF Mode** VF[15:0] Mode: 0b = VF is LAN. 1b = VF is DSI. By default, VF[15:0] are LAN.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.26.28 VSI Receive Queue Mapping Table - VSILAN_QTABLE[n,VSI] (0x00440000 + 0x1000*n + 0x4*VSI, n=0...7, VSI=0...767; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
QINDEX_0	10:0	0x0	RW	UNDEFINED	**Queue Index 0** Defines the index of the VSI queue '2*n' in the PF queues space, where 'n' is the register index. The absolute queue index in the device space equals to *QINDEX* plus PFLAN_QALLOC.*FIRSTQ* of the parent PF. Setting the *QINDEX* to 0x7FF means that the queue is not valid.
RESERVED	15:11	0x0	RSV	N/A	Reserved.
QINDEX_1	26:16	0x0	RW	UNDEFINED	**Queue Index 1** Defines the index of the VSI queue '2*n+1' in the PF queues space, where 'n' is the register index. The absolute queue index in the device space equals to *QINDEX* plus PFLAN_QALLOC.*FIRSTQ* of the parent PF. Setting the *QINDEX* to 0x7FF means that the queue is not valid.
RESERVED	31:27	0x0	RSV	N/A	Reserved.

13.2.2.26.29 VSI Queue Control - VSILAN_QBASE[VSI] (0x0044C000 + 0x4*VSI, VSI=0...767; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VSIBASE	10:0	0x0	RW	UNDEFINED	**VSI Base** Defines the base index of VSI 'n' within the range of the PF queues, where 'n' is the VSI register index. The *VSIBASE* is meaningful for this VSI only if the *VSIQTABLE_ENA* is cleared.
VSIQTABLE_ENA	11	0b	RW	UNDEFINED	**VSI Queue Table Enable** Selects between contiguous range of queues for this VSI vs. scattered range: 0b = The VSI is assigned a contiguous range starting at *VSIBASE*. 1b = The VSI is assigned a scattered range defined by the VSILAN_QTABLE.
RESERVED	31:12	0x0	RSV	N/A	Reserved.

13.2.2.27 PF - TimeSync (IEEE 1588) Registers

13.2.2.27.1 Global TimeSync Enable - GLTSYN_ENA[n] (0x00088808 + 0x4*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TSYN_ENA	0	0b	RW	UNDEFINED	**TimeSync Enable** 0b = The timer is not incremented. 1b = Enables the 1588 Master 'n', where 'n' is the register index.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.27.2 Global Master TimeSync Command - GLTSYN_CMD (0x00088810; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CMD	7:0	0x0	RW	UNDEFINED	**Command** Command to be executed by the master timer that is defined by *SEL_MASTER*. 0x01 = Init the TIME registers. 0x02 = Init the INCVAL registers. 0x03 = Init both the TIME and INCVAL registers. 0x04 = Adjust the TIME registers by the ADJ registers. 0x0C = Adjust the TIME registers after the TIME cross the INIT_TIME registers. 0x80 = Read the TIME registers of both masters to their INIT_TIME registers. All other values re reserved.
SEL_MASTER	8	0b	RW	UNDEFINED	**Select Master** 0b = Select Master 0. 1b = Select Master 1.
RESERVED	31:9	0x0	RSV	N/A	Reserved.

13.2.2.27.3 Global Master TimeSync Command SYNC Control - GLTSYN_CMD_SYNC (0x00088814; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SYNC	1:0	00b	WO	UNDEFINED	**Sync** Setting the *SYNC* field, the SYNC signals are driven to the 2 x master timers and all the PHYs for one clock, and then the field is auto-cleared by the device. The affected timers are defined by the *SEL_MASTER* in the GLTSYN_CMD register. The *SYNC* field can be one of the following options: 00b = Reserved. 01b = Increment the counter by 1 (the LS bit in the GLTSYN_TIME_L register). 10b = Decrement the counter by 1 (the LS bit in the GLTSYN_TIME_L register). 11b = Execute the programmed CMD when the SYNC signals are driven.
RESERVED	31:2	0x0	RSV	N/A	Reserved.

13.2.2.27.4 Global TimeSync Sync Delay - GLTSYN_SYNC_DLAY (0x00088818; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SYNC_DELAY	4:0	0xF	RW	UNDEFINED	**Sync Delay** Delay in 1588 clocks between assertion of the SYNC in the master register until the CMD is forwarded to the master timers. ***Note:*** This value should match the delay on the top level between the SYNC in the master register and the PHY slaves. For HH, this is the delay from SYNC assertion until the actual sampling of the timestamp.
RESERVED	31:5	0x0	RSV	N/A	Reserved.

13.2.2.27.5 Global HH Sync Delay - GLTSYN_HH_DLAY (0x0008881C; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SYNC_DELAY	3:0	0x0	RW	UNDEFINED	**Sync Delay** Delay in 1588 clocks between assertion of the SYNC in the master register until the CMD is forwarded to the master timers. ***Note:*** This value should match the delay on the top level between the SYNC in the master register and the PHY slaves. For HH, this is the delay from SYNC assertion until the actual sampling of the timestamp.
RESERVED	31:4	0x0	RSV	N/A	Reserved.

13.2.2.27.6 Global TimeSync Semaphore - PFTSYN_SEM (0x00088880; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
BUSY	0	0b	RCW	UNDEFINED	**Busy** This flag is used as a semaphore indication between software entities on the same PF or multiple PFs. Reading the *BUSY* flag by software, the bit is auto-set to '1', indicating to other software entities that the hardware resource is busy. Software clears this flag to '0' when the hardware resource is no longer needed.
RESERVED	3:1	000b	RSV	N/A	Reserved.
PF_OWNER	6:4	000b	RO	N/A	**PF Owner** When the *BUSY* flag is set, this field holds the PF that currently owns the logic.
RESERVED	31:7	0x0	RSV	N/A	Reserved.

13.2.2.27.7 Global TimeSync Status 0 - GLTSYN_STAT[n] (0x000888C0 + 0x4*n, n=0...1; RCW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
EVENT0	0	0b	RCW	UNDEFINED	**Event 0** Set to 1b when the EVENT register zero that belongs to the 1588 master 'n' = GLTSYN_EVNT[n,0] captures the time of an input event, where 'n' is the register index.
EVENT1	1	0b	RCW	UNDEFINED	**Event 1** Set to 1b when the EVENT register one that belongs to the 1588 master 'n' = GLTSYN_EVNT[n,1] captures the time of an input event, where 'n' is the register index.
EVENT2	2	0b	RCW	UNDEFINED	**Event 2** Set to 1b when the EVENT register two that belongs to the 1588 master 'n' = GLTSYN_EVNT[n,2] captures the time of an input event, where 'n' is the register index.
RESERVED	3	0b	RSV	N/A	Reserved.
TGT0	4	0b	RCW	UNDEFINED	**Target Time 0** Set to 1b when the target time zero that belongs to the 1588 master 'n' = GLTSYN_TGT[n,0] expires, where 'n' is the register index.
TGT1	5	0b	RCW	UNDEFINED	**Target Time 1** Set to 1b when the target time one that belongs to the 1588 master 'n' = GLTSYN_TGT[n,1] expires, where 'n' is the register index.
TGT2	6	0b	RCW	UNDEFINED	**Target Time 2** Set to 1b when the target time two that belongs to the 1588 master 'n' = GLTSYN_TGT[n,2] expires., where 'n' is the register index.
TGT3	7	0b	RCW	UNDEFINED	**Target Time 3** Set to 1b when the target time three that belongs to the 1588 master 'n' = GLTSYN_TGT[n,3] expires, where 'n' is the register index.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.27.8 Global TimeSync Time Zero - GLTSYN_TIME_0[n] (0x000888C8 + 0x4*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TSYNTIME_0	31:0	0x0	RW	UNDEFINED	**TimeSync Time 0** Bits 0...31 of the 96-bit master timer 'n', where 'n' is the register index. In many applications (depending on the 1588 clock and the INCVAL of the timer), this register holds the sub-nanosecond units of the timer.

13.2.2.27.9 Global TimeSync Time Low - GLTSYN_TIME_L[n] (0x000888D0 + 0x4*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TSYNTIME_L	31:0	0x0	RW	UNDEFINED	**TimeSync Time Low** Bits 32...63 of the 96-bit master timer 'n', where 'n' is the register index. In many applications (depending on the 1588 clock and the INCVAL of the timer) this register holds the 32 LS bits of the time in nanosecond units.

13.2.2.27.10 Global TimeSync Time High - GLTSYN_TIME_H[n] (0x000888D8 + 0x4*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TSYNTIME_H	31:0	0x0	RW	UNDEFINED	**TimeSync Time High** Upper 32 bits of the master timer 'n', where 'n' is the register index.

13.2.2.27.11 Global TimeSync Shadow Time Zero - GLTSYN_SHTIME_0[n] (0x000888E0 + 0x4*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TSYNTIME_0	31:0	0x0	RW	UNDEFINED	**TimeSync Time 0** Used for one of the following programming actions of the master timer 'n', where 'n' is the register index. • Init value for the GLTSYN_TIME_0 register at SYNC. • Used as time limit for master adjust action. • Used to sample the GLTSYN_TIME_0 register at SYNC.

13.2.2.27.12 Global TimeSync Shadow Time Low - GLTSYN_SHTIME_L[n] (0x000888E8 + 0x4*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TSYNTIME_L	31:0	0x0	RW	UNDEFINED	**TimeSync Time Low** Used for one of the following programming actions of the master timer 'n', where 'n' is the register index: • Init value for the GLTSYN_TIME_L register at SYNC. • Used as time limit for master adjust action. • Used to sample the GLTSYN_TIME_L register at SYNC.

13.2.2.27.13 Global TimeSync Shadow Time High - GLTSYN_SHTIME_H[n] (0x000888F0 + 0x4*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TSYNTIME_H	31:0	0x0	RW	UNDEFINED	**TimeSync Time High** Used for one of the following programming actions of the master timer 'n', where 'n' is the register index: • Init value for the GLTSYN_TIME_H register at SYNC. • Used as time limit for master adjust action. • Used to sample the GLTSYN_TIME_H register at SYNC.

13.2.2.27.14 Global TimeSync HH Time Low - GLTSYN_HHTIME_L[n] (0x000888F8 + 0x4*n, n=0...1; RO)

Field definitions are the same as those defined in Section 13.2.2.27.28.

13.2.2.27.15 Global TimeSync HH Time High - GLTSYN_HHTIME_H[n] (0x00088900 + 0x4*n, n=0...1; RO)

Field definitions are the same as those defined in Section 13.2.2.27.29.

13.2.2.27.16 Global TimeSync Shadow Adjust Low - GLTSYN_SHADJ_L[n] (0x00088908 + 0x4*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
ADJUST_L	31:0	0x0	RW	UNDEFINED	**Adjust Low** Low 32 bits of the SHADJ register. It can be used as the lower 32-bit init value of the GLTSYN_INCVAL or the lower 32 bits of the adjust by 'N' command. When used for adjust by 'N' command, negative numbers are presented in 2's complement format.

13.2.2.27.17 Global TimeSync Shadow Adjust High - GLTSYN_SHADJ_H[n] (0x00088910 + 0x4*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
ADJUST_H	31:0	0x0	RW	UNDEFINED	**Adjust High** High 32 bits of the SHADJ register. It can be used as the lower 32-bit init value of the GLTSYN_INCVAL or the lower 32 bits of the adjust by 'N' command. When used for adjust by 'N' command, negative numbers are presented in 2's complement format.

13.2.2.27.18 Global TimeSync Increment Value Low - GLTSYN_INCVAL_L[n] (0x00088918 + 0x4*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
INCVAL_L	31:0	0x0	RW	UNDEFINED	**Increment Value Low** 32 LS bits of the Increment Value added to the 96-bit 1588 master timer for each 1588 clock of the master timer 'n', where 'n' is the register index.

13.2.2.27.19 Global TimeSync Increment Value High - GLTSYN_INCVAL_H[n] (0x00088920 + 0x4*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
INCVAL_H	7:0	0x0	RW	UNDEFINED	**Increment Value Low** 8 MS bits of the Increment Value added to the 96-bit 1588 master timer for each 1588 clock.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.27.20 Global TimeSync Target Time Low - GLTSYN_TGT_L_0[n] (0x00088928 + 0x4*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TSYNTGTT_L	31:0	0x0	RW	UNDEFINED	**TimeSync Target Time Low** 32 LS bits of the target time 'm' of an event out in one of the AUX IO signals of the master timer 'n', where 'n', 'm' are the register indexes.

13.2.2.27.21 Global TimeSync Target Time High - GLTSYN_TGT_H_0[n] (0x00088930 + 0x4*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TSYNTGTT_H	31:0	0x0	RW	UNDEFINED	**TimeSync Target Time High** 32 MS bits of the target time 'm' of an event out in one of the AUX IO signals of the master timer 'n', where 'n', 'm' are the register indexes.

13.2.2.27.22 Global TimeSync Target Time Low - GLTSYN_TGT_L_1[n] (0x00088938 + 0x4*n, n=0...1; RW)

Field definitions are the same as those defined in Section 13.2.2.27.20.

13.2.2.27.23 Global TimeSync Target Time High - GLTSYN_TGT_H_1[n] (0x00088940 + 0x4*n, n=0...1; RW)

Field definitions are the same as those defined in Section 13.2.2.27.21.

13.2.2.27.24 Global TimeSync Target Time Low - GLTSYN_TGT_L_2[n] (0x00088948 + 0x4*n, n=0...1; RW)

Field definitions are the same as those defined in Section 13.2.2.27.20.

13.2.2.27.25 Global TimeSync Target Time High - GLTSYN_TGT_H_2[n] (0x00088950 + 0x4*n, n=0...1; RW)

Field definitions are the same as those defined in Section 13.2.2.27.21.

13.2.2.27.26 Global TimeSync Target Time Low - GLTSYN_TGT_L_3[n] (0x00088958 + 0x4*n, n=0...1; RW)

Field definitions are the same as those defined in Section 13.2.2.27.20.

13.2.2.27.27 Global TimeSync Target Time High - GLTSYN_TGT_H_3[n] (0x00088960 + 0x4*n, n=0...1; RW)

Field definitions are the same as those defined in Section 13.2.2.27.21.

13.2.2.27.28 Global TimeSync Event Time Low - GLTSYN_EVNT_L_0[n] (0x00088968 + 0x4*n, n=0...1; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TSYNEVNT_L	31:0	0x0	RO	N/A	**TimeSync Event Low** 32 LS bits of the sampled event 'm' time. The sampled event is defined by *EVNTLVL* field in the PRTTSYN_AUX register of the master timer 'n', where 'n', 'm' are the register indexes.

13.2.2.27.29 Global TimeSync Event Time High - GLTSYN_EVNT_H_0[n] (0x00088970 + 0x4*n, n=0...1; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TSYNEVNT_H	31:0	0x0	RO	N/A	**TimeSync Event Low** 32 MS bit of the sampled event 'm' time of a 1588 event defined by the PRTTSYN_AUX register of the master timer 'n', where 'n', 'm' are the register indexes.

13.2.2.27.30 Global TimeSync Event Time Low - GLTSYN_EVNT_L_1[n] (0x00088978 + 0x4*n, n=0...1; RO)

Field definitions are the same as those defined in Section 13.2.2.27.28.

13.2.2.27.31 Global TimeSync Event Time High - GLTSYN_EVNT_H_1[n] (0x00088980 + 0x4*n, n=0...1; RO)

Field definitions are the same as those defined in Section 13.2.2.27.29.

13.2.2.27.32 Global TimeSync Event Time Low - GLTSYN_EVNT_L_2[n] (0x00088988 + 0x4*n, n=0...1; RO)

Field definitions are the same as those defined in Section 13.2.2.27.28.

13.2.2.27.33 Global TimeSync Event Time High - GLTSYN_EVNT_H_2[n] (0x00088990 + 0x4*n, n=0...1; RO)

Field definitions are the same as those defined in Section 13.2.2.27.29.

13.2.2.27.34 Global TimeSync AUX Output Control - GLTSYN_AUX_OUT_0[n] (0x00088998 + 0x4*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
OUT_ENA	0	0b	RW	UNDEFINED	**Output Enable** Synchronized output enablement for the matched TGT register of master timer 'n'. When set to 1b, the synchronized output signal is enabled according to the other parameters in this register.
OUTMOD	2:1	00b	RW	UNDEFINED	**Output Mode** Output signal mode of operation: 00b = Output Level Mode 01b = Flipped Output Mode 10b = Output Pulse Mode 11b = Output Clock Mode The GPIO signals should be set as 1588 output by the GLGEN_GPIO_CTL[n] registers.
OUTLVL	3	0b	RW	UNDEFINED	**Output Level** Output level driven on the IO signal at the Target Time.
INT_ENA	4	0b	RW	UNDEFINED	**Interrupt Enable** At 1b, interrupt is enabled for this event.
RESERVED	7:5	000b	RSV	N/A	Reserved.
PULSEW	11:8	0x0	RW	UNDEFINED	**Pulse Width** Output pulse width for "Output Pulse Mode" equals to 16 x (PULSEW + 1) 1588 clocks.
RESERVED	31:12	0x0	RSV	N/A	Reserved.

13.2.2.27.35 Global TimeSync AUX Output Control - GLTSYN_AUX_OUT_1[n] (0x000889A0 + 0x4*n, n=0...1; RW)

Field definitions are the same as those defined in Section 13.2.2.27.34.

13.2.2.27.36 Global TimeSync AUX Output Control - GLTSYN_AUX_OUT_2[n] (0x000889A8 + 0x4*n, n=0...1; RW)

Field definitions are the same as those defined in Section 13.2.2.27.34.

13.2.2.27.37 Global TimeSync AUX Output Control - GLTSYN_AUX_OUT_3[n] (0x000889B0 + 0x4*n, n=0...1; RW)

Field definitions are the same as those defined in Section 13.2.2.27.34.

13.2.2.27.38 Global TimeSync Clock Out Duration - GLTSYN_CLKO_0[n] (0x000889B8 + 0x4*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TSYNCLKO	31:0	0x0	RW	UNDEFINED	**TimeSync Clock Out** Clock output duration 'm' defined by the units of the GLTSYN_TIME_L register, controlled by master timer 'n' as described in Section 9.7.6.1, where 'n', 'm' are the register indexes.

13.2.2.27.39 Global TimeSync Clock Out Duration - GLTSYN_CLKO_1[n] (0x000889C0 + 0x4*n, n=0...1; RW)

Field definitions are the same as those defined in Section 13.2.2.27.38.

13.2.2.27.40 Global TimeSync Clock Out Duration - GLTSYN_CLKO_2[n] (0x000889C8 + 0x4*n, n=0...1; RW)

Field definitions are the same as those defined in Section 13.2.2.27.38.

13.2.2.27.41 Global TimeSync Clock Out Duration - GLTSYN_CLKO_3[n] (0x000889D0 + 0x4*n, n=0...1; RW)

Field definitions are the same as those defined in Section 13.2.2.27.38.

13.2.2.27.42 Global TimeSync AUX Input Control - GLTSYN_AUX_IN_0[n] (0x000889D8 + 0x4*n, n=0...1; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
EVNTLVL	1:0	00b	RW	UNDEFINED	**Event Level** Event level on the I/O signal configured as 1588 input for the matched EVENT register of master timer 'n'. It can be set to one of the following options: 00b = Disable 01b = Rising Edge 10b = Falling Edge 11b = Any Transition The GPIO signals should be set as 1588 input by the GLGEN_GPIO_CTL[n] registers.
RESERVED	3:2	00b	RSV	N/A	Reserved.
INT_ENA	4	0b	RW	UNDEFINED	**Interrupt Enable** At 1b, interrupt is enabled for this event.
RESERVED	31:5	0x0	RSV	N/A	Reserved.

13.2.2.27.43 Global TimeSync AUX Input Control - GLTSYN_AUX_IN_1[n] (0x000889E0 + 0x4*n, n=0...1; RW)

Field definitions are the same as those defined in Section 13.2.2.27.42.

13.2.2.27.44 Global TimeSync AUX Input Control - GLTSYN_AUX_IN_2[n] (0x000889E8 + 0x4*n, n=0...1; RW)

Field definitions are the same as those defined in Section 13.2.2.27.42.

13.2.2.27.45 Global Hammock Harbor Timer Control - GLHH_ART_CTL (0x000A41D4; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
ACTIVE	0	0b	WO	UNDEFINED	**Active** The Firmware sets the *ACTIVE* flag to start the Sync procedure with the ART. It is then cleared by the hardware at sequence completion.
TIME_OUT1	1	0b	RO	N/A	**Timeout 1** The Local Sync message is not received from the ART within time limit.
TIME_OUT2	2	0b	RO	N/A	**Timeout 2** The Sync Comp message is not received from the ART within time limit.
RESERVED	31:3	0x0	RSV	N/A	Reserved.

13.2.2.27.46 Global Hammock Harbor ART Time High - GLHH_ART_TIME_H (0x000A41D8; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
ART_TIME_H	31:0	0x0	RO	N/A	**ART Time High** Upper 32 bits of ART time collected in the Local Sync message.

13.2.2.27.47 Global Hammock Harbor ART Time Low - GLHH_ART_TIME_L (0x000A41DC; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
ART_TIME_L	31:0	0x0	RO	N/A	**ART Time Low** Lower 32 bits of ART time collected in the Local Sync message.

13.2.2.27.48 Global Hammock Harbor Sync-Start DATA - GLHH_ART_DATA (0x000A41E0; RO)

This register defines the HH parameters. It is restricted and should not be modified by the Software or the firmware.

Field	Bit(s)	Init.	Type	CFG Policy	Description
AGENT_TYPE	2:0	001b	RW	UNDEFINED	**Agent Type** 001b for Free-Running type.
SYNC_TYPE	3	1b	RW	UNDEFINED	**Sync Type** 1b for SyncCTR based == Type D HH.
MAX_DELAY	7:4	0x8	RW	UNDEFINED	**Max Allow Delay** 0x8 for Round Range Max.
TIME_BASE	11:8	0x1	RW	UNDEFINED	**Time Base** 0x1 for 1 ns units.
RESERVED	31:12	0x0	RW	UNDEFINED	Reserved. Must be set to zero.

13.2.2.27.49 Global Hammock Harbor Semaphore - PFHH_SEM (0x000A4200; RW)

Field definitions are the same as those defined in Section 13.2.2.27.6.

13.2.2.28 PF - Protocol Engine Registers

Registers related to protocol engine functionality.

13.2.2.28.1 Protocol Engine VF CQP Doorbell - VFPE_CQPDB[VF] (0x00500000 + 0x4*VF, VF=0...255; RW)

This register is used to post work to the Protocol Engine Control QP. Software can determine if CQP has pending work by comparing *WQHEAD* to *WQTAIL* after reading the CQPTAIL register. Software must first populate one or more WQEs in the CQP WQ and then put the index of the WQE following the last populated WQE into *WQHEAD* to submit work to CQP.

Field definitions are the same as those defined in Section 13.2.2.28.3.

13.2.2.28.2 Protocol Engine VF CQP Tail - VFPE_CQPTAIL[VF] (0x00500400 + 0x4*VF, VF=0...255; RO)

This register is used to determine how much work the Protocol Engine Control QP has pending. Software can determine if CQP has pending work by comparing the last value written to *WQHEAD* in the CQPDB register to *WQTAIL* after reading this register. This register is updated after the CQP operation is complete.

Field definitions are the same as those defined in Section 13.2.2.28.4.

13.2.2.28.3 Protocol Engine CQP Doorbell - PFPE_CQPDB (0x00500800; RW)

This register is used to post work to the Protocol Engine Control QP. Software can determine if CQP has pending work by comparing *WQHEAD* to *WQTAIL* after reading the CQPTAIL register. Software must first populate one or more WQEs in the CQP WQ and then put the index of the WQE following the last populated WQE into *WQHEAD* to submit work to CQP.

Field	Bit(s)	Init.	Type	CFG Policy	Description
WQHEAD	10:0	0x0	RW	UNDEFINED	**Work Queue Head** Indicates the WQE index of the next WQE that software will post to CQP.
RESERVED	31:11	0x0	RSV	N/A	Reserved.

13.2.2.28.4 Protocol Engine CQP Tail - PFPE_CQPTAIL (0x00500880; RO)

This register is used to determine how much work the Protocol Engine Control QP has pending. Software can determine if CQP has pending work by comparing the last value written to *WQHEAD* in the CQPDB register to *WQTAIL* after reading this register. This register is updated after the CQP operation is complete.

Field	Bit(s)	Init.	Type	CFG Policy	Description
WQTAIL	10:0	0x0	RW	UNDEFINED	**Work Queue Tail** Indicates the WQE index of the next WQE that CQP will process.
RESERVED	30:11	0x0	RSV	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
CQP_OP_ERR	31	0b	RW	UNDEFINED	**CQP Operation Error** Indicates that CQP encountered an error processing an operation. If software has multiple requests outstanding to CQP at the time of the error, *WQTAIL* might not indicate the WQE that caused the error.

13.2.2.28.5 Protocol Engine VF CQ Arm - VFPE_CQARM[VF] (0x00502000 + 0x4*VF, VF=0...255; RW)

This register is used to arm a Protocol Engine Completion Queue for events in conjunction with the Completion Queue Doorbell Shadow Area located in host memory. Arming is also frequently referred to as requesting notification for a Completion Queue. Events can be generated when the next completion is generated or when the next completion related to a solicited operation is generated. On read, this register returns the value of 0.

Field definitions are the same as those defined in Section 13.2.2.28.8.

Note: This register is also located in the Protocol Engine doorbell page section of the BAR.

13.2.2.28.6 Protocol Engine VF CQ Ack - VFPE_CQACK[VF] (0x00502400 + 0x4*VF, VF=0...255; RW)

This register is used to acknowledge a completion event for a Protocol Engine Completion Queue. The interrupt processing logic that handles completion events must write this register to enable new events for a completion queue.

Field definitions are the same as those defined in Section 13.2.2.28.9.

13.2.2.28.7 Protocol Engine VF AEQ Allocate - VFPE_AEQALLOC[VF] (0x00502800 + 0x4*VF, VF=0...255; RW)

This register is used to return Asynchronous Event Queue entries back to the hardware for usage.

Field definitions are the same as those defined in Section 13.2.2.28.10.

Note: 16 instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.28.8 Protocol Engine CQ Arm - PFPE_CQARM (0x00502C00; RW)

This register is used to arm a Protocol Engine Completion Queue for events in conjunction with the Completion Queue Doorbell Shadow Area located in host memory. Arming is also frequently referred to as requesting notification for a Completion Queue. Events can be generated when the next completion is generated or when the next completion related to a solicited operation is generated. On read, this register returns the value of 0.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PECQID	18:0	0x0	RW	UNDEFINED	**Protocol Engine Completion Queue ID** Used to arm a Protocol Engine Completion Queue for generating events.
RESERVED	31:19	0x0	RSV	N/A	Reserved.

Note: This register is also located in the Protocol Engine doorbell page section of the BAR.

13.2.2.28.9 Protocol Engine CQ Ack - PFPE_CQACK (0x00502C80; RW)

This register is used to acknowledge a completion event for a Protocol Engine Completion Queue. The interrupt processing logic that handles completion events must write this register to enable new events for a completion queue.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PECQID	18:0	0x0	RW	UNDEFINED	**Protocol Engine Completion Queue ID** Used to enable new events for a Protocol Engine Completion Queue.
RESERVED	31:19	0x0	RSV	N/A	Reserved

13.2.2.28.10 Protocol Engine AEQ Allocate - PFPE_AEQALLOC (0x00502D00; RW)

This register is used to return Asynchronous Event Queue entries back to the hardware for usage.

Field	Bit(s)	Init.	Type	CFG Policy	Description
AECOUNT	31:0	0x0	RW	UNDEFINED	**Asynchronous Event Count** Specifies the number of Asynchronous Event Queue entries that have been processed by software and can now be reused by hardware.

Note: 16 instances of this register are implemented for this product. The remaining instances are reserved for future expansion.

13.2.2.28.11 Protocol Engine CQE Drop Count - GLPE_VFCQEDROPCNT[n] (0x00503000 + 0x4*n, n=0...31; RW1C)

This register counts the number of CQEs that are dropped due to the *Valid* bit being cleared in the CQ Context for this PF.

Field definitions are the same as those defined in Section 13.2.2.28.14.

Note: There are 16 of these registers, one per PF.

13.2.2.28.12 Protocol Engine CEQE Drop Count - GLPE_VFCEQEDROPCNT[n] (0x00503080 + 0x4*n, n=0...31; RW1C)

This register counts the number of CEQEs that are dropped due to the *Valid* bit being cleared in the CEQ Context for this PF.

Field definitions are the same as those defined in Section 13.2.2.28.15.

Note: There are 16 of these registers, one per PF.

13.2.2.28.13 Protocol Engine AEQE Drop Count - GLPE_VFAEQEDROPCNT[n] (0x00503100 + 0x4*n, n=0...31; RW1C)

This register counts the number of AEQEs that are dropped due to the *Valid* bit being cleared in the AEQ Context for this PF.

Field definitions are the same as those defined in Section 13.2.2.28.16.

Note: There are 16 of these registers, one per PF.

13.2.2.28.14 Protocol Engine CQE Drop Count - GLPE_PFCQEDROPCNT[n] (0x00503200 + 0x4*n, n=0...7; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CQEDROPCNT	15:0	0x0	RW1C	DYNAMIC	**CQE Drop Count** Counts the number of CQEs that are dropped due to the *Valid* bit being cleared in the CQ Context for this PF.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

Note: There are 16 of these registers, one per PF.

13.2.2.28.15 Protocol Engine CEQE Drop Count - GLPE_PFCEQEDROPCNT[n] (0x00503220 + 0x4*n, n=0...7; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CEQEDROPCNT	15:0	0x0	RW1C	DYNAMIC	**CEQE Drop Count** Counts the number of CEQEs that are dropped due to the *Valid* bit being cleared in the CEQ Context for this PF.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

Note: There are 16 of these registers, one per PF.

13.2.2.28.16 Protocol Engine AEQE Drop Count - GLPE_PFAEQEDROPCNT[n] (0x00503240 + 0x4*n, n=0...7; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
AEQEDROPCNT	15:0	0x0	RW1C	DYNAMIC	**AEQE Drop Count** Counts the number of AEQEs that are dropped due to the *Valid* bit being cleared in the AEQ Context for this PF.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

Note: There are 16 of these registers, one per PF.

13.2.2.28.17 Protocol Engine CQM Func Invalidate Register - GLPE_CQM_FUNC_INVALIDATE (0x00503300; RO)

This register is used to invalidate all CQs and related structures internal to PE pertaining to a specific function.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PF_NUM	2:0	000b	RW	UNDEFINED	**PF Number** Specifies the PF associated with the invalidate command.
VM_VF_NUM	12:3	0x0	RW	UNDEFINED	**VM/VF Number** Specifies the VM or VF associated with the invalidate command.
VM_VF_TYPE	14:13	00b	RW	UNDEFINED	**VM/VF Type** Specifies the VM_VF TYPE of the function associated with the invalidate command.
PMF_ID	20:15	0x0	RW	UNDEFINED	**PMF ID** The PMF to invalidate. This field is only valid for the doorbell array.
RESERVED	28:21	0x0	RSV	N/A	Reserved.
INVALIDATE_TYPE	30:29	00b	RW	UNDEFINED	**Invalidate Type** Specifies what is being invalidated. 00b = Invalidate CQ Cache. 01b = Invalidate doorbell array. 10b = Invalidate CQ Cache and doorbell array. 11b = Reserved.
ENABLE	31	0b	RW	UNDEFINED	**Enable** This bit is set by software to start the invalidate process, and is cleared by hardware when the invalidate process is complete.

13.2.2.28.18 Protocol Engine VF WQE Allocate Register - VFPE_WQEALLOC[VF] (0x00504000 + 0x4*VF, VF=0...255; RW)

This register is used to post work the Protocol Engine Queue Pairs. On read, this register returns the value of 0.

Field definitions are the same as those defined in Section 13.2.2.28.19.

Note: This register is also located in the Protocol Engine doorbell page section of the BAR.

13.2.2.28.19 Protocol Engine WQE Allocate Register - PFPE_WQEALLOC (0x00504400; RW)

This register is used to post work the Protocol Engine Queue Pairs. On read, this register returns the value of 0.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PEQPID	17:0	0x0	RW	UNDEFINED	**Protocol Engine Queue Pair ID** This field should be used to notify hardware that new work is available to be processed.
RESERVED	19:18	00b	RSV	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
WQE_DESC_INDEX	31:20	0x0	RW	UNDEFINED	**WQE Descriptor Index** Indicates a high 12 bits of the WQE Index in the Send Queue. This field is used to identify a Push Mode message in the head of the SQ. Push Mode message is processed by hardware only if it is in the head of the Send Queue, and otherwise it is dropped and processed when WQE is re-fetched during regular transmit operation. This field is used for the Push Mode Doorbells only. For the regular Doorbells, this field should be treated as reserved, and ignored by hardware.

Note: This register is also located in the Protocol Engine doorbell page section of the BAR.

13.2.2.28.20 Protocol Engine VF Create CQP Status - VFPE_CCQPSTATUS[VF] (0x00508000 + 0x4*VF, VF=0...255; RO)

This register is used to indicate the progress of creating the control QP for a given PCI function. During host software initialization, these bits are initially 0. Each bit is set when the associated step of the initialization process completes.

Field definitions are the same as those defined in Section 13.2.2.28.27.

13.2.2.28.21 Protocol Engine VF Create CQP Low - VFPE_CCQPLOW[VF] (0x00508400 + 0x4*VF, VF=0...255; RW)

This register stores the lower 32 bits of the 64-bit physical address of the Control QP context for its associated PCI function. Under host software control, each PCI function uses its Create Control QP High/Low registers to create its corresponding Control QP. The 64-bit address must always be updated by writing the GLPE_CCQPLOW field last. Writing a 0 to both GLPE_CCQPHIGH and GLPE_CCQLOW destroys the CQP and clears the GLPE_CCQPSTATUS.*CCQP_DONE* bit.

Field definitions are the same as those defined in Section 13.2.2.28.28.

13.2.2.28.22 Protocol Engine VF Create CQP High - VFPE_CCQPHIGH[VF] (0x00508800 + 0x4*VF, VF=0...255; RW)

This register stores the upper 32 bits of the 64-bit physical address of the Control QP context for its associated PCI function.

Field definitions are the same as those defined in Section 13.2.2.28.29.

13.2.2.28.23 Protocol Engine VF IP Config 0 - VFPE_IPCONFIG0[VF] (0x00508C00 + 0x4*VF, VF=0...255; RW)

This register is used to set or view the *IPID* field that the Protocol Engine writes into the IP header. The Protocol Engine increments this value for each outgoing IP datagram.

Field definitions are the same as those defined in Section 13.2.2.28.30.

13.2.2.28.24 Protocol Engine VF CQP Error Codes - VFPE_CQPERRCODES[VF] (0x00509000 + 0x4*VF, VF=0...255; RO)

This register reports errors encountered by CQP when CQ0 is not available. The contents of this register are only valid when the associated CCQPSTATUS.CCQP_ERR bit or CQPTAIL.CQP_ERR bit is set.

Field definitions are the same as those defined in Section 13.2.2.28.31.

13.2.2.28.25 Protocol Engine VF TCP Now Timer - VFPE_TCPNOWTIMER[VF] (0x00509400 + 0x4*VF, VF=0...255; RO)

This register is a readable register, which contains *TCP_NOW*, a 32-bit counter that provides the TCP time measurement for the all of the timers. It is also used to calculate the TS Value sent in the TCP timestamp option, *TCP_NOW* is added to *TSVAL_TICK_DELTA* to form the TS Value.

Field definitions are the same as those defined in Section 13.2.2.28.32.

13.2.2.28.26 Protocol Engine VF MRTE Index Mask - VFPE_MRTEIDXMASK[VF] (0x00509800 + 0x4*VF, VF=0...255; RO)

This register is used to change the number of significant bits to be used as the Memory Region Table index. The maximum number of bits that may be used is 22, and the minimum number of bits is 14. The number of bits should never be larger than the number of HMC MRTE objects defined for the HMC PM function associated with the PCI function. The remaining bits of the driver portion of the STag are randomized by the driver.

Field definitions are the same as those defined in Section 13.2.2.28.33.

13.2.2.28.27 Protocol Engine Create CQP Status - PFPE_CCQPSTATUS (0x0050A000; RO)

This register is used to indicate the progress of creating the control QP for a given PCI function. During host software initialization, these bits are initially 0. Each bit is set when the associated step of the initialization process completes.

Field	Bit(s)	Init.	Type	CFG Policy	Description
CCQP_DONE	0	0b	RW	UNDEFINED	**Create CQP Done** 0b = Indicates that CQP has not been created. 1b = Indicates that the Create CQP operation triggered by writing to PECCQPHIGH and PECCQPLOW has completed.
RESERVED	3:1	000b	RSV	N/A	Reserved.
HMC_PROFILE	6:4	000b	RW	UNDEFINED	**HMC Profile** Specifies the HMC resource profile that is active. 000b = Reserved 001b = Default 010b = SR-IOV VF Primary 011b = SR-IOV Even Distribution All other values are reserved.
RESERVED	15:7	0x0	RSV	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMA_EN_VFS	21:16	0x0	RW	UNDEFINED	**RDMA Enabled VFs** Specifies the number of RDMA enabled VFs allocated in the HMC Resource Profile.
RESERVED	30:22	0x0	RSV	N/A	Reserved.
CCQP_ERR	31	0b	RW	UNDEFINED	**Create CQP Error** Indicates that CQP encountered an error processing the last Create CQP request. This bit is reset when CQP is destroyed.

13.2.2.28.28 Protocol Engine Create CQP Low - PFPE_CCQPLOW (0x0050A080; RW)

This register stores the lower 32 bits of the 64-bit physical address of the Control QP context for its associated PCI function. Under host software control, each PCI function uses its Create Control QP High/Low registers to create its corresponding Control QP. The 64-bit address must always be updated by writing the GLPE_CCQPLOW field last. Writing a 0 to both GLPE_CCQPHIGH and GLPE_CCQLOW destroys the CQP and clears the GLPE_CCQPSTATUS.*CCQP_DONE* bit.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PECCQPLOW	31:0	0x0	RW	UNDEFINED	**Protocol Engine Create CQP Low** Least significant bits of the Control QP context physical address in host memory.

13.2.2.28.29 Protocol Engine Create CQP High - PFPE_CCQPHIGH (0x0050A100; RW)

This register stores the upper 32 bits of the 64-bit physical address of the Control QP context for its associated PCI function.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PECCQPHIGH	31:0	0x0	RW	UNDEFINED	**Protocol Engine Create CQP High** Most significant bits of the Control QP context physical address in host memory.

13.2.2.28.30 Protocol Engine IP Config 0 - PFPE_IPCONFIG0 (0x0050A180; RW)

This register is used to set or view the *IPID* field that the Protocol Engine writes into the IP header. The Protocol Engine increments this value for each outgoing IP datagram.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PEIPID	15:0	0x0	RW	UNDEFINED	**Protocol Engine IP ID** Specifies the IP Identification field used for IPv4 IP Header Generation. This register is initialized by firmware or driver, and incremented by hardware with each IPv4 datagram transmitted by PE both UDA and iWARP on the given PCI function.

Field	Bit(s)	Init.	Type	CFG Policy	Description
USEENTIREIDRANGE	16	0b	RW	UNDEFINED	**Use Entire ID Range** Specifies that the Protocol Engine should use the entire 16-bit range for the IPID value. When this bit is not set, only the lower 15 bits are used, and the upper bit is set to the value in PEIPID[15].
UDP_SRC_PORT_MASK_EN	17	0b	RW	UNDEFINED	**UDP Source Port Mask Enable** When this bit is set (enabled), for RoCE v2 QPs on the associated function, the *UDP Source Port* field is interpreted as a entropy mask, except if *UDP Source Port* is 0x0 (copied directly to packet in this case). For every "0" bit in the *UDP Source Port* field, the respective UDP packet header bit is randomized based on associated Destination QP ID and Destination IP Address (from Address Handle). For every "1" bit in the *UDP Source Port* field, the respective bit is copied directly as-is to the UDP packet header. This allows for selecting up to any 15 bits of UDP source port for entropy inclusion in the packet. The intent is to enable UDP ECMP capability in the network fabric, where supported in connecting equipment. When this bit is clear (disabled), the *UDP Source Port* field is simply copied as-is per software QP configuration to the UDP packet header.
RESERVED	31:18	0x0	RSV	N/A	Reserved.

13.2.2.28.31 Protocol Engine CQP Error Codes - PFPE_CQPERRCODES (0x0050A200; RO)

This register reports errors encountered by CQP when CQ0 is not available. The contents of this register are only valid when the associated CCQPSTATUS.*CCQP_ERR* bit or CQPTAIL.*CQP_ERR* bit is set.

Field	Bit(s)	Init.	Type	CFG Policy	Description
CQP_MINOR_CODE	15:0	0x0	RW	UNDEFINED	**CQP Minor Code** Minor code that would have been reported in a CQP Completion.
CQP_MAJOR_CODE	31:16	0x0	RW	UNDEFINED	**CQP Major Code** Major code that would have been reported in a CQP Completion.

13.2.2.28.32 Protocol Engine TCP Now Timer - PFPE_TCPNOWTIMER (0x0050A280; RO)

This register is a readable register, which contains *TCP_NOW*, a 32-bit counter that provides the TCP time measurement for the all of the timers. It is also used to calculate the TS Value sent in the TCP timestamp option, *TCP_NOW* is added to *TSVAL_TICK_DELTA* to form the TS Value.

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCP_NOW	31:0	0x0	RO	N/A	**TCP Now** Current value of TCP_NOW.

13.2.2.28.33 Protocol Engine MRTE Index Mask - PFPE_MRTEIDXMASK (0x0050A300; RO)

This register is used to change the number of significant bits to be used as the Memory Region Table index. The maximum number of bits that may be used is 22, and the minimum number of bits is 14. The number of bits should never be larger than the number of HMC MRTE objects defined for the HMC PM function associated with the PCI function. The remaining bits of the driver portion of the STag are randomized by the driver.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MRTEIDXMASKBITS	4:0	0x0	RW	UNDEFINED	**MRTE Index Mask Bits** Specifies the number of bits to be used for the MRTE index from the driver portion of the STag. The remaining bits (24 - *MRTEIDEMASKBITS*) are randomized by the driver. The minimum value for this field is 14 and the maximum is 22. Values outside of this range are normalized to 14 or 22 by the hardware.
RESERVED	31:5	0x0	RSV	N/A	Reserved.

13.2.2.28.34 Protocol Engine TCP Now 50us Count - GLPE_VFTCPNOW50USCNT[n] (0x0050B300 + 0x4*n, n=0...31; RO)

Tracks the number of TCPNOW ticks which occur in 50 μs.

Field definitions are the same as those defined in Section 13.2.2.28.39.

13.2.2.28.35 Protocol Engine FLM XMIT Allocate Error - GLPE_VFFLMXMITALLOCERR[n] (0x0050B400 + 0x4*n, n=0...31; RO)

This register holds a count of the XMIT free list allocation errors.

Field definitions are the same as those defined in Section 13.2.2.28.40.

13.2.2.28.36 Protocol Engine FLM Q1 Allocate Error - GLPE_VFFLMQ1ALLOCERR[n] (0x0050B480 + 0x4*n, n=0...31; RO)

This register holds a count of the Q1 free list allocation errors.

Field definitions are the same as those defined in Section 13.2.2.28.41.

13.2.2.28.37 Protocol Engine FLM Read Response Allocate Error - GLPE_VFFLMRRFALLOCERR[n] (0x0050B500 + 0x4*n, n=0...31; RO)

This register holds a count of the Read Response free list allocation errors.

Field definitions are the same as those defined in Section 13.2.2.28.42.

13.2.2.28.38 Protocol Engine FLM Out of Order Send Completion (OOISC) Allocate Error - GLPE_VFFLMOOISCALLOCERR[n] (0x0050B580 + 0x4*n, n=0...31; RO)

This register holds a count of the Read Response free list allocation errors.

Field definitions are the same as those defined in Section 13.2.2.28.43.

13.2.2.28.39 Protocol Engine TCP Now 50us Count - GLPE_PFTCPNOW50USCNT[n] (0x0050B8C0 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
CNT	31:0	0x0	RW	UNDEFINED	**Count** Tracks the number of TCPNOW ticks which occur in 50 μs.

13.2.2.28.40 Protocol Engine FLM XMIT Allocate Error - GLPE_PFFLMXMITALLOCERR[n] (0x0050B900 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
ERROR_COUNT	15:0	0x0	RO	N/A	**Error Count** The number of failed free list allocations for this function. Counter does not roll over. Software read resets.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.28.41 Protocol Engine FLM Q1 Allocate Error - GLPE_PFFLMQ1ALLOCERR[n] (0x0050B920 + 0x4*n, n=0...7; RO)

This register holds a count of the Q1 free list allocation errors.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ERROR_COUNT	15:0	0x0	RO	N/A	**Error Count** The number of failed free list allocations for this function. Counter does not roll over. Software read resets.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.28.42 Protocol Engine FLM Read Response Allocate Error - GLPE_PFFLMRRFALLOCERR[n] (0x0050B940 + 0x4*n, n=0...7; RO)

This register holds a count of the Read Response free list allocation errors.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ERROR_COUNT	15:0	0x0	RO	N/A	**Error Count** The number of failed free list allocations for this function. Counter does not roll over. Software read resets.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.28.43 Protocol Engine FLM Out of Order Send Completion (OOISC) Allocate Error - GLPE_PFFLMOOISCALLOCERR[n] (0x0050B960 + 0x4*n, n=0...7; RO)

This register holds a count of the Read Response free list allocation errors.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ERROR_COUNT	15:0	0x0	RO	N/A	**Error Count** The number of failed free list allocations for this function. Counter does not roll over. Software read resets.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.28.44 Protocol Engine CPU Status 0 - GLPE_CPUSTATUS0 (0x0050BA5C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PECPUSTATUS0	31:0	0x0	RW	UNDEFINED	**Protocol Engine CPU Status 0** Provides the Host with the current status of one of the Protocol Engine internal CPUs. Typically, the exclusive writer of this register is CQP, and the exclusive reader is Host software, but other usage models are supported. ***Note:*** Details on the meaning of each status code are not defined in this specification and may change with firmware revision.

13.2.2.28.45 Protocol Engine CPU Status 1 - GLPE_CPUSTATUS1 (0x0050BA60; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PECPUSTATUS1	31:0	0x0	RW	UNDEFINED	**Protocol Engine CPU Status 1** Provides the Host with the current status of one of the Protocol Engine internal CPUs. Typically, the exclusive writer of this register is TEP, and the exclusive reader is Host software, but other usage models are supported. ***Note:*** Details on the meaning of each status code are not defined in this specification and may change with firmware revision.

13.2.2.28.46 Protocol Engine CPU Status 2 - GLPE_CPUSTATUS2 (0x0050BA64; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PECPUSTATUS2	31:0	0x0	RW	UNDEFINED	**Protocol Engine CPU Status 2** Provides the Host with the current status of one of the Protocol Engine internal CPUs. Typically, the exclusive writer of this register is OOP, and the exclusive reader is Host software, but other usage models are supported. ***Note:*** Details on the meaning of each status code are not defined in this specification and may change with firmware revision.

13.2.2.28.47 PEPM Control - GLPE_PEPM_CTRL (0x0050C000; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PEPM_ENABLE	0	0b	RW	UNDEFINED	**PEPM Enable** 0b = PEPM is disabled, pepm_fc is driven to 1's, and interfaces are blocked/NAK'ed. 1b = PEPM is enabled, pepm_fc behaves normally.
RESERVED	7:1	0x0	RSV	N/A	Reserved.
PEPM_HALT	8	0b	RW	UNDEFINED	**PEPM Halt** 0b = PEPM is not halted, pepm_fc behaves normally. 1b = PEPM is halted, pepm_fc bits driven to '0', and no credits allocated or returned.
RESERVED	15:9	0x0	RSV	N/A	Reserved.
PEPM_PUSH_MARGIN	23:16	0x25	RW	UNDEFINED	**PEPM Push Margin** Used to calculate when to NAK a PUSH request. This value is subtracted from the *PQID* threshold value before comparing to the incoming MDQ credit request.
RESERVED	31:24	0x0	RSV	N/A	Reserved.

13.2.2.28.48 PEPM Dealloc - GLPE_PEPM_DEALLOC (0x0050C004; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MDQ_CREDITS	13:0	0x0	RW	UNDEFINED	**MDQ Credits** The number of MDQ credits to deallocate. Must be non-zero if *PSQ_CREDITS*=0.
PSQ_CREDITS	18:14	0x0	RW	UNDEFINED	**PSQ Credits** The number of PSQ credits to deallocate. Must be non-zero if *MDQ_CREDITS*=0.
PQID	27:19	0x0	RW	UNDEFINED	**PQ ID** The PQ ID associated with the credits to deallocate. Must be <= max number of PQs supported for the current configuration.
PORT	30:28	000b	RW	UNDEFINED	**Port** The port associated with the credits to deallocate. Must be <= the max number of ports supported for the current configuration.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DEALLOC_RDY	31	0b	SC	UNDEFINED	**Deallocate Ready** This bit should be set to initiate a request to deallocate the number of MDQ/PSQ credits indicated in this register. This bit is cleared by hardware to indicate that the credits have been deallocated. This register must never be changed when the *DEALLOC_RDY* bit is set, or your results will be indeterminate.

13.2.2.28.49 PEPM PSQ Count - GLPE_PEPM_PSQ_COUNT (0x0050C020; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PEPM_PSQ_COUNT	15:0	0x0	RW	UNDEFINED	**PEPM PSQ Count** Indicates the number of allocated PSQ credits per port.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.28.50 PEPM PSQ/MDQ Count - PRT_PEPM_COUNT[n] (0x0050C040 + 0x4*n, n=0...511; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PEPM_PSQ_COUNT	4:0	0x0	RW	UNDEFINED	**PEPM PSQ Count** The current count for the number of entries on the PSQ (Packet Scheduling Queue) for the PQ. This value is incremented by allocate requests and decremented by deallocate requests.
RESERVED	15:5	0x0	RSV	N/A	Reserved.
PEPM_MDQ_COUNT	29:16	0x0	RW	UNDEFINED	**PEPM MDQ Count** The current count for the number of entries on the MDQ (Metadata Queue) for the PQ. This value is incremented by allocate requests and decremented by deallocate requests.
RESERVED	31:30	0x0	RSV	N/A	Reserved.

13.2.2.28.51 PEPM PQ Threshold - GLPE_PEPM_THRESH[n] (0x0050C840 + 0x4*n, n=0...511; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PEPM_PSQ_THRESH	4:0	0x0	RW	UNDEFINED	**PEPM PSQ Threshold** The threshold for the PSQ (Packet Scheduling Queue) for the PQ. This value is compared against the *PSQ_COUNT* to determine if the allocation request is successful.
RESERVED	15:5	0x0	RSV	N/A	Reserved.
PEPM_MDQ_THRESH	29:16	0x0	RW	UNDEFINED	**PEPM MDQ Threshold** The threshold for the MDQ (Metadata Queue) for the PQ. This value is compared against the *MDQ_COUNT* to determine if the allocation request is successful.
RESERVED	31:30	0x0	RSV	N/A	Reserved.

13.2.2.28.52 PE Push PEPM - GLPE_PUSH_PEPM (0x0053241C; RO)

Credit values for Push requests from PEPM.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MDQ_CREDITS	7:0	0x29	RW	UNDEFINED	**MDQ Credits** The number of credits Push requests from PEPM for a Push WQE doorbell.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.28.53 MDQ Base - GLPE_MDQ_BASE[n] (0x00536000 + 0x4*n, n=0...511; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MDOC_INDEX	27:0	0x0	RW	UNDEFINED	**MDOC Index** MDOC base index for this Metadata Queue. The MDQ starts at this index.
RESERVED	31:28	0x0	RSV	N/A	Reserved.

13.2.2.28.54 MDQ Size - GLPE_MDQ_SIZE[n] (0x00536800 + 0x4*n, n=0...511; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MDQ_SIZE	13:0	0x0	RW	UNDEFINED	**MDQ Size** Size of MDQ for this PQ. The MDQ wraps at index=base+size.
RESERVED	31:14	0x0	RSV	N/A	Reserved.

13.2.2.28.55 MDQ Pointer - GLPE_MDQ_PTR[n] (0x00537000 + 0x4*n, n=0...511; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MDQ_HEAD	13:0	0x0	RW	UNDEFINED	**MDQ Head** Head pointer for Metadata Queue objects in MDOC.
RESERVED	15:14	00b	RSV	N/A	Reserved.
MDQ_TAIL	29:16	0x0	RW	UNDEFINED	**MDQ Tail** Tail pointer for Metadata Queue objects in MDOC.
RESERVED	31:30	00b	RSV	N/A	Reserved.

13.2.2.29 PF - Manageability Registers

13.2.2.29.1 MNG FW RAM Status Registers - GL_MNG_FW_RAM_STAT (0x0008309C; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FW_RAM_RST_STAT	0	0b	RO	N/A	**Firmware RAM Reset State** Set by firmware and cleared when RAM is initialized: 1. POR reset. 2. EMPR followed ECC reset in EMP memories.
MNG_MEM_ECC_ERR	1	0b	RO	N/A	**MNG Memory EXX Error** ECC error in one of MNG memories. Should be cleaned by firmware.
RESERVED	31:2	0x0	RSV	N/A	Reserved.

13.2.2.29.2 Firmware Reset Count - GL_FWRESETCNT (0x00083100; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
FWRESETCNT	31:0	0x0	RO	N/A	**Firmware Resets Count** Updated by Hardware. Saturates at 0xFFFF,FFFF.

13.2.2.29.3 SHA Extend Value - GL_MNG_SHA_EXTEND[n] (0x00083120 + 0x4*n, n=0...7; RO)

Can be written (by firmware) only once per EMP reset assertion.

Field	Bit(s)	Init.	Type	CFG Policy	Description
GL_MNG_SHA_EXTEND	31:0	0x0	RO	UNDEFINED	**MNG SHA Extend** SHA value calculated on signed image.

13.2.2.29.4 SHA Extend Value Status - GL_MNG_SHA_EXTEND_STATUS (0x00083148; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
STAGE	2:0	000b	RW	UNDEFINED	**Stage** Firmware load SHA reflected in registers: 000b = No value. 001b = ROM version (in GL_MNG_SHA_EXTEND_ROM). 010b = Mini-loader SHA (in GL_MNG_SHA_EXTEND_ROM). 011b = Mini-loader SHA (no Full FW) (in GL_MNG_SHA_EXTEND_ROM). 100b = Full firmware SHA (in GL_MNG_SHA_EXTEND) and Mini-loader SHA (in GL_MNG_SHA_EXTEND_ROM). All other values are reserved.
RESERVED	29:3	0x0	RSV	UNDEFINED	Reserved.
FW_HALTED	30	0b	RW	UNDEFINED	**Firmware Halted** If set, firmware load process was halted before completion.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DONE	31	0b	RW	UNDEFINED	**Done** If set, indicates this is the final version of GL_MNG_SHA_EXTEND expected.

13.2.2.29.5 SHA ROM Extend Value - GL_MNG_SHA_EXTEND_ROM[n] (0x00083160 + 0x4*n, n=0...7; RO)

Can be written (by firmware) only once per EMP reset assertion.

Field	Bit(s)	Init.	Type	CFG Policy	Description
GL_MNG_SHA_EXTEND_ROM	31:0	0x0	RO	UNDEFINED	**MNG SHA Extend ROM** SHA value calculated on signed image.

13.2.2.29.6 Hardware Arbitration Control - GL_MNG_HWARB_CTRL (0x000B6130; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
NCSI_ARB_EN	0	0b	RW	UNDEFINED	**NC-SI Arbitration Enable** Hardware Arbitration Enable. If this bit is set, it is assumed the NCSI_ARB_IN and NCSI_ARB_OUT are connected to a Hardware arbitration ring. Otherwise, the NCSI_ARB_IN pin is pulled up internally.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.29.7 Firmware Status - GL_MNG_FWSM (0x000B6134; RO)

This register reflects the firmware load status. All the bits in the registers are RW bits and the firmware implements the functionality. Bits 15:0 are reset by EMP reset.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FW_MODES	1:0	00b	RW	UNDEFINED	**Firmware Modes** Indicate in which mode the firmware operates: 00b = Normal Mode 01b = Debug mode 10b = Recovery Mode 11b = Debug + Recovery mode.
RESERVED	9:2	0x0	RW	N/A	Reserved.
EEP_RELOAD_IND	10	0b	RW	UNDEFINED	**EEP Reload Indication** 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	N/A	Reserved.
PCIR_AL_FAILURE	16	0b	RW	UNDEFINED	**PCIR Auto-Load Failure** A PCI Reset is requested
POR_AL_FAILURE	17	0b	RW	UNDEFINED	**POR Auto-Load Failure** A Power-on Reset is requested.
RESERVED	18	0b	RW	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
EXT_ERR_IND	24:19	0x0	RW	UNDEFINED	**External Error Indication** Firmware writes here the reason that the firmware operation has stopped. Possible values are: 0x00 = No error 0x01 = Parser module failed 0x02 = Switch module failed. 0x03 = Scheduler module failed. 0x04 = DCB module failed 0x05 = Link module failed 0x06 = LLDP module failed 0x07 = Manage module failed. 0x08 = ACL module failed 0x09 = Flow director/RSS module failed 0x3E = Mini-loader failed (If set, devices is in blank flash programming.) 0x3F = ROM process failure (if set, devices is in blank flash programming.) All other values are reserved. ***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.
RESERVED	31:25	0x0	RW	N/A	Reserved.

13.2.2.29.8 General FW Debug Registers - GENERAL_MNG_FW_DBG_CSR[n] (0x000B6180 + 0x4*n, n=0...9; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
GENERAL_FW_DBG	31:0	0x0	RW	UNDEFINED	**General Firmware Debug** All-purpose general firmware debug.

13.2.2.29.9 Management Ethernet Type Filters - PRT_MNG_METF[n] (0x00214120 + 0x20*n, n=0...3; RO)

The METF registers are written by the BMC and are not accessible to the host for writing. The registers are used to filter manageability packets (see Section 12.4).

Reset - The METF registers are cleared on LAN_PWR_GOOD only. The initial values for this register might be loaded from the EEPROM after power-up reset.

Field	Bit(s)	Init.	Type	CFG Policy	Description
ETYPE	15:0	0x0	RW	UNDEFINED	**EtherType** 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	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
POLARITY	30	0b	RW	UNDEFINED	**Polarity** 0b = Positive filter - Filter enters the decision filters if a match occurred. 1b = Negative filter - Filter enters the decision filters if a match did not occur.
RESERVED	31	0b	RSV	N/A	Reserved.

13.2.2.29.10 Manageability IPv4 Address Filter - PRT_MNG_MIPAF4[n] (0x002141A0 + 0x20*n, n=0...3; RO)

The Manageability IPv6 Address Filter register stores IPv4 Addresses for manageability filtering.

Note: These registers should be written in network order.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MIPAF	31:0	0x0	RW	UNDEFINED	**Manageability IP Address Filters** For each n, m, m=0...3, n=0...3, MIPAF[m,n] register holds DWord `n' of IPv6 filter `m' (4 x IPv6 filters).

13.2.2.29.11 Manageability MAC Address High - PRT_MNG_MMAH[n] (0x00214220 + 0x20*n, n=0...3; RO)

These registers contain the upper bits of the 48-bit Ethernet address. The complete address is {MMAH,MMAL}. The MMAH registers are written by the BMC and are not accessible to the host for writing. The registers are used to filter manageability packets (see Section 12.4).

The initial values for this register can be loaded from the EEPROM after power-up reset or firmware reset.

Note: The MMAH.*MMAH* field should be written in network order.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MMAH	15:0	0x0	RW	UNDEFINED	**Manageability MAC Address High** The upper 16 bits of the 48-bit Ethernet address. ***Note:*** Appears in Big Endian order (MS byte of MMAH is last on the wire).
RESERVED	31:16	0x0	RSV	N/A	Reserved. Reads as 0. Ignored on write.

13.2.2.29.12 Manageability MAC Address Low - PRT_MNG_MMAL[n] (0x002142A0 + 0x20*n, n=0...3; RO)

These registers contain the lower bits of the 48-bit Ethernet address. The MMAL registers are written by the internal firmware and are not accessible to the host for writing. The registers are used to filter manageability packets (see Section 12.4).

Reset - The MMAL registers are cleared on LAN_PWR_GOOD only. The initial values for this register can be loaded from the EEPROM after power-up reset.

Note: The MMAH.*MMAL* field should be written in network order.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MMAL	31:0	0x0	RW	UNDEFINED	**Manageability MAC Address Low** The lower 32 bits of the 48-bit Ethernet address. ***Note:*** Appears in Big Endian order (LS byte of MMAL is first on the wire).

13.2.2.29.13 Management Flex UDP/TCP Ports - PRT_MNG_MFUTP[n] (0x00214320 + 0x20*n, n=0...15; RO)

Each 32-bit register (n=0...15) refers to one UDP/TCP port filter.

The MFUTP registers are written by the BMC and are not accessible to the host for writing. The registers are used to filter manageability packets (see Section 12.4).

Reset - The MFUTP registers are cleared on LAN_PWR_GOOD only. The initial values for this register can be loaded from the EEPROM after power-up reset.

Note: The *MFUTP_N* fields should be written in network order.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MFUTP_N	15:0	0x0	RW	UNDEFINED	**Management Flex UDP/TCP Ports n** n-th Management Flex UDP/TCP port.
UDP	16	0b	RW	UNDEFINED	**UDP** Match if port is UDP
TCP	17	0b	RW	UNDEFINED	**TCP** Match if port is TCP
SOURCE_DESTINATION	18	0b	RW	UNDEFINED	**Source/Destination** 0b = Compare Destination port. 1b = Compare Source port.
RESERVED	31:19	0x0	RSV	N/A	Reserved.

13.2.2.29.14 Manageability IPv6 Address Filter - PRT_MNG_MIPAF6[n] (0x00214520 + 0x20*n, n=0...15; RO)

The Manageability IPv6 Address Filter register stores IPv6 Addresses for manageability filtering.

Note: These registers should be written in network order.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MIPAF	31:0	0x0	RW	UNDEFINED	**Manageability IP Address Filters** For each n, m, m=0...3, n=0...3, MIPAF[m,n] register holds DWord `n' of IPv6 filter `m' (4 x IPv6 filters).

13.2.2.29.15 Management Control Register - PRT_MNG_MANC (0x00214720; RO)

The MANC register can be written by the BMC and is not accessible to the host for writing.

Field	Bit(s)	Init.	Type	CFG Policy	Description
FLOW_CONTROL_DISCARD	0	0b	RW	UNDEFINED	**Flow Control Discard** 0b = Apply filtering rules to packets with Flow Control EtherType. 1b = Discard packets with Flow Control EtherType. ***Note:*** Flow Control EtherType is 0x8808.
NCSI_DISCARD	1	0b	RW	UNDEFINED	**NC-SI Discard** 0b = Apply filtering rules to packets with NC-SI EtherType. 1b = Discard packets with NC-SI EtherType. ***Note:*** NC-SI EtherType is 0x88F8.
RESERVED	16:2	0x0	RSV	N/A	Reserved.
RCV_TCO_EN	17	0b	RW	UNDEFINED	**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	24:18	0x0	RSV	N/A	Reserved.
FIXED_NET_TYPE	25	0b	RW	UNDEFINED	**Fixed Next Type** 0b = Both tagged and un-tagged packets can be forwarded to manageability engine. 1b = Only packets matching the net type defined by the *NET_TYPE* field pass to manageability.
NET_TYPE	26	0b	RW	UNDEFINED	**Net Type** 0b = Pass only un-tagged packets. 1b = Pass only VLAN tagged packets. Valid only if *FIXED_NET_TYPE* is set.
RESERVED	27	0b	RSV	N/A	Reserved.
EN_BMC2OS	28	0b	RW	UNDEFINED	**Enable BMC-to-OS and OS-to-BMC Traffic** 0b = The BMC cannot communicate with the OS. 1b = The BMC can communicate with the OS. When cleared, the BMC traffic is not forwarded to the OS, even if the Host address filtering indicates that it should. When cleared, the OS traffic is not forwarded to the BMC, even if the manageability decision filters indicates it should. This bit does not impact the BMC to Network traffic. ***Note:*** • Initial value loaded according to value of Port n traffic types field in NVM. • Bit reflects internal management Aux register bit.

Field	Bit(s)	Init.	Type	CFG Policy	Description
EN_BMC2NET	29	0b	RW	UNDEFINED	**Enable BMC-to-network and network-to-BMC Traffic** 0b = The BMC can not communicate with the network. 1b = 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. ***Notes:*** • Initial value loaded according to value of Port n traffic types field in NVM. • Bit reflects internal management Aux register bit. • This bit can change while the host is sending or receiving traffic.
RESERVED	31:30	00b	RSV	N/A	Reserved.

13.2.2.29.16 Management Only Traffic Register - PRT_MNG_MNGONLY (0x00214740; RO)

The MNGONLY register allows exclusive filtering of certain type of traffic to the BMC. Exclusive filtering enables the BMC to define certain packets that are forwarded to the BMC but not to the host. The packets are not forwarded to the host even if they pass the host L2 filtering process.

Each manageability decision filter (MDEF and MDEF_EXT) has a corresponding bit in the MNGONLY register. When a manageability decision filter (MDEF and MDEF_EXT) forwards a packet to manageability, it might also block the packet from being forwarded to the host if the corresponding *MNGONLY* bit is set.

Field	Bit(s)	Init.	Type	CFG Policy	Description
EXCLUSIVE_TO_MANAGEABILITY	7:0	0x0	RW	UNDEFINED	**Exclusive to Manageability** 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	N/A	Reserved.

13.2.2.29.17 Manageability Special Filters Modifiers - PRT_MNG_MSFM (0x00214760; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PORT_26F_UDP	0	1b	RW	UNDEFINED	**Port 26F UDP** Port 0x26F match if protocol is UDP.
PORT_26F_TCP	1	1b	RW	UNDEFINED	**Port 26F TCP** Port 0x26F match if protocol is TCP.
PORT_298_UDP	2	1b	RW	UNDEFINED	**Port 298 UDP** Port 0x298 match if protocol is UDP.
PORT_298_TCP	3	1b	RW	UNDEFINED	**Port 298 TCP** Port 0x298 match if protocol is TCP.
IPV6_0_MASK	4	0b	RW	UNDEFINED	**IPv6 0 Mask** Compare only 24 LSB bits of IPv6 Address 0 (MIPAF[0]).

Field	Bit(s)	Init.	Type	CFG Policy	Description
IPV6_1_MASK	5	0b	RW	UNDEFINED	**IPv6 1 Mask** Compare only 24 LSB bits of IPv6 Address 1 (MIPAF[1]).
IPV6_2_MASK	6	0b	RW	UNDEFINED	**IPv6 2 Mask** Compare only 24 LSB bits of IPv6 Address 2 (MIPAF[2]).
IPV6_3_MASK	7	0b	RW	UNDEFINED	**IPv6 3 Mask** Compare only 24 LSB bits of IPv6 Address 3 (MIPAF[3]).
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.29.18 Management VLAN TAG Value - PRT_MNG_MAVTV[n] (0x00214780 + 0x20*n, n=0...7; RO)

The MAVTV registers are written by the BMC and are not accessible to the host for writing. The registers are used to filter manageability packets (see Section 12.4).

Field	Bit(s)	Init.	Type	CFG Policy	Description
VID	11:0	0x0	RW	UNDEFINED	**VLAN ID** Contains the VLAN ID that should be compared with the incoming packet inner VLAN ID if the corresponding bit in MDEF is set.
RESERVED	31:12	0x0	RSV	N/A	Reserved.

13.2.2.29.19 Manageability Decision Filters1 - PRT_MNG_MDEF[n] (0x00214880 + 0x20*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MAC_EXACT_AND	3:0	0x0	RW	UNDEFINED	**MAC Exact AND** 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), and so on.
BROADCAST_AND	4	0b	RW	UNDEFINED	**Broadcast AND** Controls the inclusion of broadcast address filtering in the manageability filter decision (AND section).
VLAN_AND	12:5	0x0	RW	UNDEFINED	**VLAN AND** Controls the inclusion of VLAN tag 0 to 7, respectively, in the manageability filter decision (AND section). Bit 5 corresponds to VLAN tag 0, and so on.
IPV4_ADDRESS_AND	16:13	0x0	RW	UNDEFINED	**IPv4 Address AND** Controls the inclusion of IPv4 Address 0 to 3, respectively, in the manageability filter decision (AND section). Bit 13 corresponds to IPv4 Address 0, and so on. ***Note:*** These bits are set also for an ARP request packet if the Target IP match the IP Address configured in the MIPAF register.
IPV6_ADDRESS_AND	20:17	0x0	RW	UNDEFINED	**IPv6 Address AND** Controls the inclusion of IPv6 Address 0 to 3, respectively, in the manageability filter decision (AND section). Bit 17 corresponds to IPv6 Address 0, and so on.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MAC_EXACT_OR	24:21	0x0	RW	UNDEFINED	**MAC Exact OR** 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), and so on.
BROADCAST_OR	25	0b	RW	UNDEFINED	**Broadcast OR** Controls the inclusion of broadcast address filtering in the manageability filter decision (OR section).
MULTICAST_AND	26	0b	RW	UNDEFINED	**Multicast AND** Controls the inclusion of Multicast Address filtering in the manageability filter decision (AND section). Broadcast packets are not included by this bit.
ARP_REQUEST_OR	27	0b	RW	UNDEFINED	**ARP Request OR** Controls the inclusion of ARP Request filtering in the manageability filter decision (OR section).
ARP_RESPONSE_OR	28	0b	RW	UNDEFINED	**ARP Response OR** Controls the inclusion of ARP Response filtering in the manageability filter decision (OR section).
NEIGHBOR_DISCOVERY_134_OR	29	0b	RW	UNDEFINED	**Neighbor Discovery 134 OR** Controls the inclusion of Neighbor Discovery filtering in the manageability filter decision (OR section). The neighbor type accepted by this filter is type 0x86 (134).
PORT_0X298_OR	30	0b	RW	UNDEFINED	**Port 0x298 OR** Controls the inclusion of port 0x298 filtering in the manageability filter decision (OR section).
PORT_0X26F_OR	31	0b	RW	UNDEFINED	**Port 0x26F OR** Controls the inclusion of port 0x26F filtering in the manageability filter decision (OR section).

13.2.2.29.20 Management Decision Filters VSI - PRT_MNG_MDEFVSI[n] (0x00214980 + 0x20*n, n=0...3; RO)

This register is used to define the VSIs that is used to receive packets that matched a specific MDEF. In case of multiple match, the VSI assigned to the MDEF with the highest index is used.

Note: In A0 the lowest index is used.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MDEFVSI_2N	15:0	0x0	RW	UNDEFINED	**MDEF VSI 2n** Defines the VSI used for packets matching MDEF 2*n.
MDEFVSI_2NP1	31:16	0x0	RW	UNDEFINED	**MDEF VSI 2n+1** Defines the VSI used for packets matching MDEF 2*n+1.

13.2.2.29.21 Manageability Decision Filters - PRT_MNG_MDEF_EXT[n] (0x00214A00 + 0x20*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
L2_ETHERTYPE_AND	3:0	0x0	RW	UNDEFINED	**L2 EtherType AND** Controls the inclusion of L2 EtherType filtering in the manageability filter decision (AND section).
L2_ETHERTYPE_OR	7:4	0x0	RW	UNDEFINED	**L2 EtherType OR** Controls the inclusion of L2 EtherType filtering in the manageability filter decision (OR section).
FLEX_PORT_OR	23:8	0x0	RW	UNDEFINED	**Flex Port OR** Controls the inclusion of Flex port filtering in the manageability filter decision (OR section). Bit 16 corresponds to flex port 0, and so on.
FLEX_TCO	24	0b	RW	UNDEFINED	**Flex TCO** Controls the inclusion of Flex TCO filtering in the manageability filter decision (OR section). Bit 24 corresponds to Flex TCO filter. ***Note:*** Supported only for Network traffic.
NEIGHBOR_DISCOVERY_135_OR	25	0b	RW	UNDEFINED	**Neighbor Discovery 135 OR** Controls the inclusion of Neighbor Discovery filtering in the manageability filter decision (OR section). The neighbor type accepted by this filter is type 0x87 (135).
NEIGHBOR_DISCOVERY_136_OR	26	0b	RW	UNDEFINED	**Neighbor Discovery 136 OR** Controls the inclusion of Neighbor Discovery filtering in the manageability filter decision (OR section). The neighbor type accepted by this filter is type 0x88 (136).
NEIGHBOR_DISCOVERY_137_OR	27	0b	RW	UNDEFINED	**Neighbor Discovery 137 OR** Controls the inclusion of Neighbor Discovery filtering in the manageability filter decision (OR section). The neighbor type accepted by this filter is type 0x89 (137).
ICMP_OR	28	0b	RW	UNDEFINED	**ICMP OR** Controls the inclusion of ICMP filtering in the manageability filter decision (OR section).
MLD	29	0b	RW	UNDEFINED	**MLD** Controls the inclusion of MLD packets. These are ICMPv6 packets with the following types: 130, 131, 132, 143.
APPLY_TO_NETWORK_TRAFFIC	30	0b	RW	UNDEFINED	**Apply to Network Traffic** 0b = This decision filter does not apply to traffic received from the network. 1b = This decision filter applies to traffic received from the network.
APPLY_TO_HOST_TRAFFIC	31	0b	RW	UNDEFINED	**Apply to Host Traffic** 0b = This decision filter does not apply to traffic received from the host. 1b = This decision filter applies to traffic received from the host.

13.2.2.29.22 Port to MDEF Set Mapping - GL_SWT_PRT2MDEF[n] (0x00216018 + 0x4*n, n=0...31; RO)

For all the registers in the manageability section that are per-port, should be per MDEF set assigned by this register to virtual ports.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MDEFIDX	2:0	000b	RW	UNDEFINED	**MDEF Index** Mapping from port to MDEF set. The selection between port and virtual port is done according to GL_SWT_SWIDFVIDX.*PORT_TYPE* field.
RESERVED	30:3	0x0	RSV	N/A	Reserved.
MDEFENA	31	0b	RW	UNDEFINED	**MDEF Enable** Enables MDEF for this port.

13.2.2.30 PF - Malicious Prevention Registers

13.2.2.30.1 Malicious VF Driver Detected on Tx TDPU - VP_MDET_TX_TDPU[VF] (0x00040000 + 0x4*VF, VF=0...255; RWC)

Malicious detect per VF. Hardware sets to '1' upon malicious event for corresponding VF. Writing '1' to this CSR clears the malicious indication.

Field definitions are the same as those defined in Section 13.2.2.30.2.

13.2.2.30.2 Malicious PF Driver Detected on Tx TDPU - PF_MDET_TX_TDPU (0x00040800; RWC)

Malicious detect per VF. Hardware sets to '1' upon malicious event for corresponding VF. Writing '1' to this CSR clears the malicious indication.

Field	Bit(s)	Init.	Type	CFG Policy	Description
VALID	0	0b	RWC	DYNAMIC	**Valid** A malicious event has been detected by the Tx Dat processing unit on this function.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.30.3 Malicious VF Driver Detected on Tx TCLAN - VP_MDET_TX_TCLAN[VF] (0x000FB800 + 0x4*VF, VF=0...255; RW1C)

Field definitions are the same as those defined in Section 13.2.2.30.4.

13.2.2.30.4 Malicious PF Driver Detected on Tx TCLAN - PF_MDET_TX_TCLAN (0x000FC000; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VALID	0	0b	RW1C	DYNAMIC	**Valid** A malicious event has been detected by the TCLAN unit on this function
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.30.5 Malicious Driver Tx Event Details - GL_MDET_TX_TCLAN (0x000FC068; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
QNUM	14:0	0x0	RO	DYNAMIC	**Queue Number** Absolute queue ID on which the event was detected.
VF_NUM	22:15	0x0	RO	DYNAMIC	**VF Number** Absolute VF number on which the event was detected.
PF_NUM	25:23	000b	RO	DYNAMIC	**PF Number** PF/parent PF number on which the event was detected.
RESERVED	30:26	0x0	RSV	N/A	Reserved.

Field	Bit(s)	Init.	Type	CFG Policy	Description
VALID	31	0b	RW1C	DYNAMIC	**Valid** Indicates that an event has been captured.

13.2.2.30.6 Malicious PF Driver Detected on Tx TCLAN - VM_MDET_TX_TCLAN[n] (0x000FC348 + 0x4*n, n=0...767; RW1C)

Field definitions are the same as those defined in Section 13.2.2.30.4.

13.2.2.30.7 RLAN Malicious Events - GLRLAN_MDET (0x00294200; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	31:0	0x0	RSV	N/A	Reserved.

13.2.2.30.8 Malicious Driver Rx Checks Enabled - GL_MDCK_RX (0x0029422C; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DESC_ADDR	0	1b	RW	UNDEFINED	Enable malicious event: descriptor fetch failed. For proper operation this flag must be active.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.30.9 Malicious PF Driver Detected on Rx - PF_MDET_RX (0x00294280; RW1C)

This register records a malicious event detected on the Rx-Queues. Once read, driver must write 0xFFFF to clear.

Field	Bit(s)	Init.	Type	CFG Policy	Description
VALID	0	0b	RW1C	DYNAMIC	**Valid** A malicious event has been detected on this function.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.30.10 Malicious VF Driver Detected on Rx - VP_MDET_RX[VF] (0x00294400 + 0x4*VF, VF=0...255; RW1C)

This register records a malicious event detected on the Rx-Queues. Once read, driver must write 0xFFFF to clear.

Field definitions are the same as those defined in Section 13.2.2.30.9.

13.2.2.30.11 Malicious Driver Rx Event Details - GL_MDET_RX (0x00294C00; RW1C)

This register records the details of the first Rx event detected.

Field	Bit(s)	Init.	Type	CFG Policy	Description
QNUM	14:0	0x0	RO	DYNAMIC	**Queue Number** Absolute queue ID on which the event was detected.
VF_NUM	22:15	0x0	RO	DYNAMIC	**VF Number** Absolute VF number on which the event was detected.
PF_NUM	25:23	000b	RO	DYNAMIC	**PF Number** PF/parent PF number on which the event was detected.
MAL_TYPE	30:26	0x0	RO	DYNAMIC	**Malicious Type** ID of the event that has been recorded.
VALID	31	0b	RW1C	DYNAMIC	**Valid** Indicates that an event has been captured.

13.2.2.30.12 Malicious VF Driver Detected on Tx PQM - VP_MDET_TX_PQM[VF] (0x002D2000 + 0x4*VF, VF=0...255; RW1C)

This register records a malicious event detected on the Tx-Queues. Once read, driver must write 0xFFFF to clear.

Field definitions are the same as those defined in Section 13.2.2.30.13.

13.2.2.30.13 Malicious PF Driver Detected on Tx PQM - PF_MDET_TX_PQM (0x002D2C80; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
VALID	0	0b	RW1C	DYNAMIC	**Valid** A malicious event has been detected by the PQM unit on this function.
RESERVED	31:1	0x0	RSV	N/A	Reserved.

13.2.2.30.14 Malicious Driver Tx Command Checks PQM Configuration 1 - GL_MDCK_CFG1_TX_PQM (0x002D2DF4; RW)

This register includes configuration which are used by PQM malicious detection (hardware) checks.

Field	Bit(s)	Init.	Type	CFG Policy	Description
SSO_MAX_DATA_LEN	7:0	0xFF	RW	UNDEFINED	**SSO Max Data Length** Max Data Length in 64-bytes resolution up to 16K-64 bytes.
SSO_MAX_PKT_CNT	13:8	0x3F	RW	UNDEFINED	**SSO Max Packet Count** Max number of packets incorporated in the Quanta.
RESERVED	15:14	00b	RSV	N/A	Reserved.
SSO_MAX_DESC_CNT	21:16	0x3F	RW	UNDEFINED	**SSO Max Descriptor Count** Max number of descriptor incorporated in the Quanta.
RESERVED	31:22	0x0	RSV	N/A	Reserved.

13.2.2.30.15 Malicious Driver Tx Command Checks Enable PQM - GL_MDCK_EN_TX_PQM (0x002D2DFC; RW)

This register indicates which of PQM Tx malicious detection (hardware) checks are enabled.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PCI_DUMMY_COMP	0	1b	RW	UNDEFINED	**PCI Dummy Completion** Enable detection of PCI Dummy Completion (for a QD Fetch Request)
PCI_UR_COMP	1	1b	RW	UNDEFINED	**PCI Unsupported Request Completion** Enable detection of PCI "Unsupported Request" Completion (for a QD Fetch Request).
RESERVED	2	0b	RSV	N/A	Reserved.
RCV_SH_BE_LSO	3	1b	RW	UNDEFINED	Empty Q fetch (initiated by PQMMNG after 1st Quanta was delivered to PQMMNG by DBL) and LSO QD is expected in completion. Completion is received however received QD is NOT LSO.
Q_FL_MNG_EPY_CH	4	1b	RW	UNDEFINED	**Queue Empty** In fetch command (initiated by PQMMNG) force_fetch is set (Q is full PQMMNG wise) but PQMQDC concluded that the Q is empty.
Q_EPY_MNG_FL_CH	5	1b	RW	UNDEFINED	**Queue Full** In fetch command (initiated by PQMMNG) force_fetch is clear (Q is empty PQMMNG wise) but PQMQDC concludes that the Q is not empty.
LSO_NUMDESCS_ZERO	6	1b	RW	UNDEFINED	**LSO Number of Descriptors Zero** Enable detection of LSO QD whose Number of Descriptors is zero.
LSO_LENGTH_ZERO	7	1b	RW	UNDEFINED	**LSO Length Zero** Enable detection of LSO QD whose Length is zero.
LSO_MSS_BELOW_MIN	8	1b	RW	UNDEFINED	**LSO MSS Below Minimum** Enable detection of LSO QD whose MSS is below GL_MDCK_CFG2_TX_PQM.*LSO_MIN_MSS*.
LSO_MSS_ABOVE_MAX	9	1b	RW	UNDEFINED	**LSO MSS Above Maximum** Enable detection of LSO QD whose MSS is above GL_MDCK_CFG2_TX_PQM.*LSO_MAX_MSS*.
LSO_HDR_SIZE_ZERO	10	1b	RW	UNDEFINED	**LSO Header Size Zero** Enable detection of LSO QD whose Header Size is zero.
RCV_CNT_BE_LSO	11	1b	RW	UNDEFINED	Enable detection of LSO QD for a Q which is LSO disabled.
SKIP_ONE_QT_ONLY	12	1b	RW	UNDEFINED	**Skip One Quanta Only** The first Quanta of a LSO QD that is composed of multiple Quantas is delivered to hardware. The first Quanta is scheduled and hardware fetches LSO QD so it can schedule the other Quantas of the same LSO QD. During processing of the fetched LSO QD, hardware concludes it is a single Quanta (which has already been scheduled) although DBL suggested it is multiple Quantas.
LSO_PKTCNT_ZERO	13	1b	RW	UNDEFINED	**LSO Packet Count Zero** Enable detection of LSO DBL whose First Quanta Number of Segments is zero.
SSO_LENGTH_ZERO	14	1b	RW	UNDEFINED	**SSO Length Zero** Enable detection of SSO QD whose Length is zero.
SSO_LENGTH_EXCEED	15	1b	RW	UNDEFINED	**SSO Length Exceeded** Enable detection of SSO QD whose Length exceeds GL_MDCK_CFG1_TX_PQM.*SSO_MAX_DATA_LEN*.

Field	Bit(s)	Init.	Type	CFG Policy	Description
SSO_PKTCNT_ZERO	16	1b	RW	UNDEFINED	**SSO Packet Count Zero** Enable detection of SSO QD whose Packet Count is zero.
SSO_PKTCNT_EXCEED	17	1b	RW	UNDEFINED	**SSO Packet Count Exceeded** Enable detection of SSO QD whose Packet Count exceeds GL_MDCK_CFG1_TX_PQM.*SSO_MAX_PKT_CNT*.
SSO_NUMDESCS_ZERO	18	1b	RW	UNDEFINED	**SSO Number of Descriptors Zero** Enable detection of SSO QD whose Number of Descriptors is zero.
SSO_NUMDESCS_EXCEED	19	1b	RW	UNDEFINED	**SSO Number of Descriptors Exceeded** Enable detection of SSO QD whose Number of Descriptors exceed minimum of {Remain-Ring-Length and GL_MDCK_CFG1_TX_PQM.*SSO_MAX_DESC_CNT*}.
TAIL_GT_RING_LENGTH	20	1b	RW	UNDEFINED	**Tail Greater Than Ring Length** Enable detection of DBL whose Tail is greater than Ring Length. Applicable for both QD and Legacy Interfaces.
RESERVED_DBL_TYPE	21	1b	RW	UNDEFINED	**Reserved Doorbell Type** Enable detection of DBL whose *Type* field is Reserved (2'b11), which is considered malicious.
ILLEGAL_HEAD_DROP_DBL	22	1b	RW	UNDEFINED	**Illegal Head Drop Doorbell** Enable detection of Head Drop DBL over Comms Queue for which Head Drop is disabled.
LSO_OVER_COMMS_Q	23	1b	RW	UNDEFINED	**LSO Over Comms Queue** Enable detection of LSO DBL over Comms Queue for which LSO is disabled.
ILLEGAL_VF_QNUM	24	1b	RW	UNDEFINED	**Illegal VF Queue Number** Enable detection of VF-TYPE DBLQ element that is associated with illegal Q number.
QTAIL_GT_RING_LENGTH	25	1b	RW	UNDEFINED	**Queue Tail Greater Than Ring Length** Enable detection of DBL whose QDTail is greater than Ring Length. Applicable for QD Interface.
RESERVED	31:26	0x0	RSV	N/A	Reserved.

13.2.2.30.16 Malicious Driver Tx Event Details PQM - GL_MDET_TX_PQM (0x002D2E00; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PF_NUM	2:0	000b	RO	DYNAMIC	**PF Number** PF/parent PF number on which the event was detected.
RESERVED	3	0b	RSV	N/A	Reserved
VF_NUM	11:4	0x0	RO	DYNAMIC	**VF Number** Absolute VF number on which the event was detected.
QNUM	25:12	0x0	RO	DYNAMIC	**Queue Number** Absolute queue ID on which the event was detected.
RESERVED	30:26	0x0	RSV	N/A	Reserved.
VALID	31	0b	RW1C	DYNAMIC	**Valid** Indicates that an event has been captured.

13.2.2.31 PF - Rx QoS Registers

Rx QoS configuration registers.

13.2.2.31.1 DCB Receive RDMA Pipe Monitor Status - PRTDCB_RRDMAPMS (0x00122120; RW)

Meaningless when GLDCB_RSPMC.*RPM_MODE* is not set to 00b.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RDMARPPM	17:0	0x0	RW	UNDEFINED	**RDMA Receive Per-Port Pipe Monitor** Current amount of bytes accounted by the Per-Port Rx RDMA Pipe Monitor. It is expressed in byte units of Layer2 packet lengths, including preamble, IPG, and CRC. Writing to this field has the effect of loading a new current value into the Rx RDMA Pipe Monitor count, which is used by Rx-ETS algorithm.
RESERVED	31:18	0x0	RSV	N/A	Reserved.

13.2.2.31.2 DCB Receive per Port Pipe Monitor Control - PRTDCB_RPPMC (0x00122240; RW)

Meaningless when GLDCB_RSPMC.*RPM_MODE* is not set to 00b.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	31:0	0x0	RSV	N/A	Reserved.

13.2.2.31.3 DCB Receive Pacing Control - GLDCB_RPCC (0x00122260; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RESERVED	31:0	0x0	RSV	N/A	Reserved.

13.2.2.31.4 DCB Receive LAN Pipe Monitor Status - PRTDCB_RLANPMS (0x00122280; RW)

Meaningless when GLDCB_RSPMC.*RPM_MODE* is not set to 00b.

Field	Bit(s)	Init.	Type	CFG Policy	Description
LANRPPM	17:0	0x0	RW	UNDEFINED	**LAN Receive Per-Port Pipe Monitor** Current amount of bytes accounted by the Per-Port Rx LAN Pipe Monitor. It is expressed in byte units of Layer2 packet lengths, including preamble, IPG, and CRC. Writing to this field has the effect of loading a new current value into the Rx LAN Pipe Monitor count, which is used by Rx-ETS algorithm.
RESERVED	31:18	0x0	RSV	N/A	Reserved.

13.2.2.31.5 DCB Receive per TC PFC Timer Queue - GLDCB_RTCTQ[n] (0x001222C0 + 0x4*n, n=0...31; RO)

One register per UP. Register index corresponds to TCID.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RXQNUM	10:0	0x0	RO	N/A	**Rx-Queue Number** Returns the Rx-Queue number (which had not enough available Rx-Descriptors) that caused the PFCTIMER of the Port/TC to time out. The queue index reported in this register is the absolute queue index in the device space, which is different than the queue index used for the Tx-Queue and Rx-Queue registers. The value is meaningful only when the corresponding bit in PFDCB_RUPTI is set.
RESERVED	15:11	0x0	RSV	N/A	Reserved.
IS_PF_Q	16	0b	RO	UNDEFINED	**Is PF Queue** Set to 1b if the queue belongs to a PF.
RESERVED	31:17	0x0	RSV	UNDEFINED	Reserved.

13.2.2.31.6 DCB Receive per TC PFC Timer Status - GLDCB_RTCTS[n] (0x00122340 + 0x4*n, n=0...31; RW)

One register per UP. Register index corresponds to TCID.

Field	Bit(s)	Init.	Type	CFG Policy	Description
PFCTIMER	13:0	0x0	RW	UNDEFINED	**PFC Timer** Current value of the count-down PFC Timer of TC n in Rx, where n is the register index in the array. The amount is expressed in milliseconds. Writing to this field has the effect of loading a new current timer value, which can be useful for diagnostic purposes.
RESERVED	31:14	0x0	RSV	N/A	Reserved.

13.2.2.31.7 DCB Receive Shared Pipe Monitor Status - GLDCB_RSPMS (0x001223C0; RW)

Meaningless when GLDCB_RSPMC.*RPM_MODE* is not set to 00b.

Field	Bit(s)	Init.	Type	CFG Policy	Description
RSPM	17:0	0x0	RW	UNDEFINED	**Receive Shared Pipe Monitor** Current amount to bytes accounted by the Rx Shared Pipe Monitor. It is expressed in byte units of Layer2 packet lengths, including preamble, IPG, and CRC. Writing to this field has the effect of loading a new current value into the Rx Shared Pipe Monitor count, which is used by Rx-ETS algorithm. It is for debugging purposes only and must be avoided during normal operations.
RESERVED	31:18	0x0	RSV	N/A	Reserved.

13.2.2.31.8 DCB Receive Shared Pipe Monitor Control - GLDCB_RSPMC (0x001223C4; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RSPM	7:0	0x70	RW	UNDEFINED	**Receive Shared Pipe Monitor** Depth of the shared Pipe Monitor applied over the Rx-Pipe, for all ports and LAN/RDMA pipes altogether. It is expressed in KB units of Layer2 packet lengths, including preamble, IPG, and CRC. Meaningful only when *RPM_MODE* is set to 01b. Max allowed value = 0xEF. Unused by SWR/RPRS (used only in RCB instance of this CSR).
RPM_MODE	9:8	0x0	RW	UNDEFINED	**Receive Pipe Monitor Mode** 00b = Per Port Pipe Monitor. 01b = Shared Pipe Monitor. 10b = No Pipe Monitor. Pipes are filled up to their maximum capacity. 11b = Reserved.
PRR_MAX_EXP	13:10	0xB	RW	UNDEFINED	**Port Round-Robin Maximum Exponent** Maximum Exponent M used to compute the Max Credits that a Port can accumulate. Max Credits = (2^M) x PRTDCB_RPRRC.*BWSHARE*.
PFCTIMER	27:14	0x800	RW	UNDEFINED	**PFC Timer** Value in milliseconds loaded into the per-UP PFC Timer in Rx. The timer is loaded every time the device waits for software to free Rx-Descriptors before dropping a packet destined to a no-drop Rx-Queue. There is one such timer per UP and per port, which all are loaded with the same value. Unused by SWR/RPRS (used only in RCB instance of this CSR).
RESERVED	30:28	0x0	RSV	N/A	Reserved.
RPM_DIS	31	0b	RW	UNDEFINED	**Rx-Pipe Monitor(s) Disable** This bit controls all the Rx-Pipe monitors. 0b = 0b - Enabled 1b = Disabled Unused by SWR/RPRS (used only in RCB instance of this CSR).

13.2.2.31.9 DCB Receive Manageability Pipe Monitor Control - GLDCB_RMPMC (0x001223C8; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RSPM	5:0	0x9	RW	UNDEFINED	**Receive Shared Pipe Monitor** Depth of the Manageability Pipe Monitor applied over the Rx-Pipe, for all ports and LAN/RDMA pipes altogether. It is expressed in KB units of Layer2 packet lengths, including preamble, IPG, and CRC. It monitors all the bytes in transit between RPB and the shared 24 KB manageability input queue (included), but excluding the frame currently added into the pipe. Default setting is 9 KB = trunc(12 KB - 2.2 KB), assuming max frame size is for FCoE. Meaningful only when *RMPM_DIS* is set to 0b. Max allowed value = 0xEF.

Field	Bit(s)	Init.	Type	CFG Policy	Description
MIQ_NODROP_MODE	10:6	0x0	RW	UNDEFINED	**Manageability Input Queue No-Drop Mode** This is a bitmap. When bit n is set to 1b, Manageability Input Queue n is operated in no-drop mode. Any manageability packet fetched from RPB that is destined to this queue is NOT be dropped until it has reached the EMP. ***Note:*** Manageability packets mapped to a PB queue that is operated in drop mode can still be dropped if the RPB queue is congested. When bit n is set to 0b, Manageability Input Queue n is operated in drop mode. Manageability packets fetched from RPB that are destined to this queue can be dropped under congestion conditions.
RESERVED	30:11	0x0	RSV	N/A	Reserved.
RPM_DIS	31	0b	RW	UNDEFINED	**Rx-Pipe Monitor Disable** 0b = Enabled 1b = Disabled

13.2.2.31.10 DCB Receive Manageability Pipe Monitor Status - GLDCB_RMPMS (0x001223CC; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
RMPM	15:0	0x0	RO	N/A	**Receive Manageability Pipe Monitor** Current amount to bytes accounted by the Rx Manageability Pipe Monitor. It is expressed in byte units of Layer2 packet lengths, including preamble, IPG, and CRC.
RESERVED	31:16	0x0	RSV	N/A	Reserved.

13.2.2.31.11 DCB Receive per TC PFC Timer Indication - GLDCB_RTCTI (0x001223D0; RW1C)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PFCTIMEOUT_TC	31:0	0x0	RW1C	DYNAMIC	**PFC Time-out TCs** Bitmap where a bit set to 1b means that the PFC Timer corresponding to the Port/TC has timed out. For up to 4 link topology: Bits 0-7 are for port 0, TC 0 to 7. Bits 8-15 are for port 1, TC 0 to 7. Bits 16-23 are for port 2, TC 0 to 7. Bits 24-31 are for port 3, TC 0 to 7. For more than 4 link topology: Bits 00-03 are for port 0, TC 0 to 3. Bits 04-07 are for port 1, TC 0 to 3. Bits 08-11 are for port 2, TC 0 to 3. Bits 12-15 are for port 3, TC 0 to 3. Bits 16-19 are for port 4, TC 0 to 3. Bits 20-23 are for port 5, TC 0 to 3. Bits 24-27 are for port 6, TC 0 to 3. Bits 28-31 are for port 7, TC 0 to 3. Writing a bit with 1b restarts the corresponding PFC Timer.

13.2.2.31.12 RCB Configuration Change on the Fly Counter - GLRCB_CFG_COTF_CNT[n] (0x001223D4 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MRKR_COTF_CNT	5:0	0x0	ROCV	N/A	**Marker Change on-the-Fly Counter** Counts cfg_cotf markers originating on same configuration. Counter is incremented when cfg_cotf marker of configuration index correspond to counter index is selected by DCB arbiter in RCB and dropped. Counter is cleared on read.
RESERVED	31:6	0x0	RSV	N/A	Reserved.

13.2.2.31.13 RCB Configuration Change on the Fly Status - GLRCB_CFG_COTF_ST (0x001223F4; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MRKR_COTF_ST	7:0	0x0	RO	N/A	**Marker Change on-the-Fly Status** Bit i reflects the status of the counter. 0b = Counter is either idle or has not yet reached threshold. 1b = Counter has reached threshold and is waiting to be cleared. Bit i is set when the correspond counter reaches threshold. Bit i is cleared when the correspond counter is read.
RESERVED	31:8	0x0	RSV	N/A	Reserved.

13.2.2.31.14 Rx PM Dedicated Pool Size - GLRPRS_PMCFG_DPS[n] (0x00200308 + 0x4*n, n=0...15; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
DPS	19:0	0xFF	RW	UNDEFINED	**Dedicated Pool Size** It is expressed in bytes or commands according to usage.
RESERVED	31:20	0x0	RSV	N/A	Reserved.

13.2.2.31.15 Rx PM Dedicated Pool High Watermark - GLRPRS_PMCFG_DHW[n] (0x00200388 + 0x4*n, n=0...15; RO)

One register per TC. Register index corresponds to TCID.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DHW	19:0	0xFF	RW	UNDEFINED	**Dedicated Pool High Watermark** It is expressed in bytes or commands according to usage.
RESERVED	31:20	0x0	RSV	N/A	Reserved.

13.2.2.31.16 Rx PM Dedicated Pool Low Watermark - GLRPRS_PMCFG_DLW[n] (0x002003C8 + 0x4*n, n=0...15; RO)

One register per TC. Register index corresponds to TCID.

Field	Bit(s)	Init.	Type	CFG Policy	Description
DLW	19:0	0xFF	RW	UNDEFINED	**Dedicated Pool Low Watermark** It is expressed in bytes or commands according to usage.
RESERVED	31:20	0x0	RSV	N/A	Reserved.

13.2.2.31.17 Rx PM Shared Pool Size - GLRPRS_PMCFG_SPS[n] (0x00200408 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SPS	19:0	0xFF	RW	UNDEFINED	**Shared Pool Size** Number of bytes/commands allocated to the shared pool.
RESERVED	31:20	0x0	RSV	N/A	Reserved.

13.2.2.31.18 Rx PM Shared Pool High Watermark - GLRPRS_PMCFG_SHW[n] (0x00200448 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SHW	19:0	0xFF	RW	UNDEFINED	**Shared Pool High Watermark** It is expressed in bytes or commands according to usage.
RESERVED	31:20	0x0	RSV	N/A	Reserved.

13.2.2.31.19 Rx PM Shared Pool Low Watermark - GLRPRS_PMCFG_SLW[n] (0x00200468 + 0x4*n, n=0...7; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
SLW	19:0	0xFF	RW	UNDEFINED	**Shared Pool Low Watermark** It is expressed in bytes or commands according to usage.
RESERVED	31:20	0x0	RSV	N/A	Reserved.

13.2.2.31.20 TC Pool Config - GLRPRS_PMCFG_TC_CFG[n] (0x00200488 + 0x4*n, n=0...31; RO)

Field definitions are the same as those defined in Section 13.2.2.31.23.

13.2.2.31.21 Rx PM TC High Watermark - GLRPRS_PMCFG_TCHW[n] (0x00200588 + 0x4*n, n=0...31; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCHW	19:0	0xFF	RW	UNDEFINED	**TC High Watermark** It is expressed in bytes or commands according to usage.
RESERVED	31:20	0x0	RSV	N/A	Reserved.

13.2.2.31.22 Rx PM TC Low Watermark - GLRPRS_PMCFG_TCLW[n] (0x00200608 + 0x4*n, n=0...31; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
TCLW	19:0	0xFF	RW	UNDEFINED	**TC Low Watermark** It is expressed in bytes or commands according to usage.
RESERVED	31:20	0x0	RSV	N/A	Reserved.

13.2.2.31.23 TC Pool Config - GLSWT_PMCFG_TC_CFG[n] (0x00204900 + 0x4*n, n=0...31; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
D_POOL	3:0	0x0	RW	UNDEFINED	**Dedicated Pool** Index of dedicated pool TC belongs to.
RESERVED	15:4	0x0	RSV	N/A	Reserved.
S_POOL	18:16	000b	RW	UNDEFINED	**Shared Pool** Index of shared pool TC belongs to.
RESERVED	31:19	0x0	RSV	N/A	Reserved.

13.2.3 BAR3 Registers Summary

13.2.3.1 PF - MSI-X Table Registers Summary

Table 13-41. PF - MSI-X Table Registers Summary

Offset / Alias Offset	Abbreviation	Name	Section Reference
0x00000000 + 0x10*n, n=0...2047	MSIX_TADD[n]	MSI-X Message Address Low	13.2.4.1.1
0x00000004 + 0x10*n, n=0...2047	MSIX_TUADD[n]	MSI-X Message Address High	13.2.4.1.2
0x00000008 + 0x10*n, n=0...2047	MSIX_TMSG[n]	MSI-X Message Data	13.2.4.1.3
0x0000000C + 0x10*n, n=0...2047	MSIX_TVCTRL[n]	MSI-X Vector Control	13.2.4.1.4
0x00008000 + 0x4*n, n=0...63	MSIX_PBA[n]	MSI-X PBA Structure	13.2.4.1.5

13.2.4 Detailed Register Descriptions - PF BAR3

13.2.4.1 PF - MSI-X Table Registers

13.2.4.1.1 MSI-X Message Address Low - MSIX_TADD[n] (0x00000000 + 0x10*n, n=0...2047; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MSIXTADD10	1:0	00b	RW	UNDEFINED	**Message Address 1:0** 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	UNDEFINED	**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.

13.2.4.1.2 MSI-X Message Address High - MSIX_TUADD[n] (0x00000004 + 0x10*n, n=0...2047; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MSIXTUADD	31:0	0x0	RW	UNDEFINED	**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.

13.2.4.1.3 MSI-X Message Data - MSIX_TMSG[n] (0x00000008 + 0x10*n, n=0...2047; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MSIXTMSG	31:0	0x0	RW	UNDEFINED	**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.

13.2.4.1.4 MSI-X Vector Control - MSIX_TVCTRL[n] (0x0000000C + 0x10*n, n=0...2047; RW)

Field	Bit(s)	Init.	Type	CFG Policy	Description
MASK	0	1b	RW	UNDEFINED	**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).
RESERVED	31:1	0x0	RSV	N/A	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.

13.2.4.1.5 MSI-X PBA Structure - MSIX_PBA[n] (0x00008000 + 0x4*n, n=0...63; RO)

Field	Bit(s)	Init.	Type	CFG Policy	Description
PENBIT	31:0	0x0	RO	N/A	**MSI-X Pending Bits** Each bit is set to 1b when the appropriate interrupt request is set, and cleared to 0b when the appropriate interrupt request is cleared.

13.3 Device Registers - VF

13.3.1 VF Registers Mapping in the PF Space

Table 13-42. VF Registers Mapping in the PF Space

Abbreviation	Virtual Address	Physical Address
VFGEN_RSTAT	0x00008800	0x00074000 + 0x4*VF, VF=0...255
PFPCI_VF_FLUSH_DONE	0x0000E400	0x0009E000 + 0x4*VF, VF=0...255
VFINT_ITRN	0x00002800 + 0x4*n + 0x40*m, n=0...15, m=0...2	0x00154000 + 0x2000*n + 0x4*INT, n=0...2, INT=0...2047
VFINT_ITRN_64	0x00002C00 + 0x4*n + 0x100*m, n=0...63, m=0...2	0x00154000 + 0x2000*n + 0x4*INT, n=0...2, INT=0...2047
VFINT_DYN_CTLN	0x00003800 + 0x4*n, n=0...63	0x00160000 + 0x4*INT, INT=0...2047
VFINT_ITR0	0x00004C00 + 0x4*n, n=0...2	0x00154000 + 0x2000*n + 0x4*INT, n=0...2, INT=0...2047
VFINT_DYN_CTL0	0x00005C00	0x00160000 + 0x4*INT, INT=0...2047
VF_MBX_ARQBAH	0x00006000	0x0022B800 + 0x4*VF, VF=0...255
VF_MBX_ATQH	0x00006400	0x0022AC00 + 0x4*VF, VF=0...255
VF_MBX_ATQLEN	0x00006800	0x0022A800 + 0x4*VF, VF=0...255
VF_MBX_ARQBAL	0x00006C00	0x0022B400 + 0x4*VF, VF=0...255
VF_MBX_ARQT	0x00007000	0x0022C400 + 0x4*VF, VF=0...255
VF_MBX_ARQH	0x00007400	0x0022C000 + 0x4*VF, VF=0...255
VF_MBX_ATQBAH	0x00007800	0x0022A400 + 0x4*VF, VF=0...255
VF_MBX_ATQBAL	0x00007C00	0x0022A000 + 0x4*VF, VF=0...255
VF_MBX_ARQLEN	0x00008000	0x0022BC00 + 0x4*VF, VF=0...255
VF_MBX_ATQT	0x00008400	0x0022B000 + 0x4*VF, VF=0...255
QTX_TAIL	0x00000000 + 0x4*DBQM, DBQM=0...255	0x002C0000 + 0x4*DBQM, DBQM=0...16383
QRX_TAIL	0x00002000 + 0x4*QRX, QRX=0...255	0x00290000 + 0x4*QRX, QRX=0...2047
VFPE_IPCONFIG0	0x00008C00	0x00508C00 + 0x4*VF, VF=0...255
VFPE_CCQPHIGH	0x00009800	0x00508800 + 0x4*VF, VF=0...255
VFPE_CQPERRCODES	0x00009C00	0x00509000 + 0x4*VF, VF=0...255
VFPE_CQPTAIL	0x0000A000	0x00500400 + 0x4*VF, VF=0...255
VFPE_AEQALLOC	0x0000A400	0x00502800 + 0x4*VF, VF=0...255
VFPE_TCPNOWTIMER	0x0000A800	0x00509400 + 0x4*VF, VF=0...255
VFPE_CCQPLOW	0x0000AC00	0x00508400 + 0x4*VF, VF=0...255
VFPE_CQACK	0x0000B000	0x00502400 + 0x4*VF, VF=0...255
VFPE_CQARM	0x0000B400	0x00502000 + 0x4*VF, VF=0...255
VFPE_CCQPSTATUS	0x0000B800	0x00508000 + 0x4*VF, VF=0...255
VFPE_CQPDB	0x0000BC00	0x00500000 + 0x4*VF, VF=0...255
VFPE_WQEALLOC	0x0000C000	0x00504000 + 0x4*VF, VF=0...255

Table 13-42. VF Registers Mapping in the PF Space

Abbreviation	Virtual Address	Physical Address
VF_MBX_CPM_ATQBAL	0x0000F000	0x0022C800 + 0x4*VF128, VF128=0...127
VF_MBX_CPM_ATQBAH	0x0000F010	0x0022CA00 + 0x4*VF128, VF128=0...127
VF_MBX_CPM_ATQLEN	0x0000F020	0x0022CC00 + 0x4*VF128, VF128=0...127
VF_MBX_CPM_ATQH	0x0000F030	0x0022CE00 + 0x4*VF128, VF128=0...127
VF_MBX_CPM_ATQT	0x0000F040	0x0022D000 + 0x4*VF128, VF128=0...127
VF_MBX_CPM_ARQBAL	0x0000F050	0x0022D200 + 0x4*VF128, VF128=0...127
VF_MBX_CPM_ARQBAH	0x0000F060	0x0022D400 + 0x4*VF128, VF128=0...127
VF_MBX_CPM_ARQLEN	0x0000F070	0x0022D600 + 0x4*VF128, VF128=0...127
VF_MBX_CPM_ARQH	0x0000F080	0x0022D800 + 0x4*VF128, VF128=0...127
VF_MBX_CPM_ARQT	0x0000F090	0x0022DA00 + 0x4*VF128, VF128=0...127
VF_SB_CPM_ATQBAL	0x0000F100	0x0022E800 + 0x4*VF128, VF128=0...127
VF_SB_CPM_ATQBAH	0x0000F110	0x0022EA00 + 0x4*VF128, VF128=0...127
VF_SB_CPM_ATQLEN	0x0000F120	0x0022EC00 + 0x4*VF128, VF128=0...127
VF_SB_CPM_ATQH	0x0000F130	0x0022EE00 + 0x4*VF128, VF128=0...127
VF_SB_CPM_ATQT	0x0000F140	0x0022F000 + 0x4*VF128, VF128=0...127
VF_SB_CPM_ARQBAL	0x0000F150	0x0022F200 + 0x4*VF128, VF128=0...127
VF_SB_CPM_ARQBAH	0x0000F160	0x0022F400 + 0x4*VF128, VF128=0...127
VF_SB_CPM_ARQLEN	0x0000F170	0x0022F600 + 0x4*VF128, VF128=0...127
VF_SB_CPM_ARQH	0x0000F180	0x0022F800 + 0x4*VF128, VF128=0...127
VF_SB_CPM_ARQT	0x0000F190	0x0022FA00 + 0x4*VF128, VF128=0...127
VF_MBX_HLP_ATQBAL	0x00020000	0x0022DC00 + 0x4*VF16, VF16=0...15
VF_MBX_HLP_ATQBAH	0x00020010	0x0022DC40 + 0x4*VF16, VF16=0...15
VF_MBX_HLP_ATQLEN	0x00020020	0x0022DC80 + 0x4*VF16, VF16=0...15
VF_MBX_HLP_ATQH	0x00020030	0x0022DCC0 + 0x4*VF16, VF16=0...15
VF_MBX_HLP_ATQT	0x00020040	0x0022DD00 + 0x4*VF16, VF16=0...15
VF_MBX_HLP_ARQBAL	0x00020050	0x0022DD40 + 0x4*VF16, VF16=0...15
VF_MBX_HLP_ARQBAH	0x00020060	0x0022DD80 + 0x4*VF16, VF16=0...15
VF_MBX_HLP_ARQLEN	0x00020070	0x0022DDC0 + 0x4*VF16, VF16=0...15
VF_MBX_HLP_ARQH	0x00020080	0x0022DE00 + 0x4*VF16, VF16=0...15
VF_MBX_HLP_ARQT	0x00020090	0x0022DE40 + 0x4*VF16, VF16=0...15
VF_MBX_PSM_ATQBAL	0x00021000	0x0022DE80 + 0x4*VF16, VF16=0...15
VF_MBX_PSM_ATQBAH	0x00021010	0x0022DEC0 + 0x4*VF16, VF16=0...15
VF_MBX_PSM_ATQLEN	0x00021020	0x0022DF00 + 0x4*VF16, VF16=0...15
VF_MBX_PSM_ATQH	0x00021030	0x0022DF40 + 0x4*VF16, VF16=0...15
VF_MBX_PSM_ATQT	0x00021040	0x0022DF80 + 0x4*VF16, VF16=0...15
VF_MBX_PSM_ARQBAL	0x00021050	0x0022DFC0 + 0x4*VF16, VF16=0...15
VF_MBX_PSM_ARQBAH	0x00021060	0x0022E000 + 0x4*VF16, VF16=0...15
VF_MBX_PSM_ARQLEN	0x00021070	0x0022E040 + 0x4*VF16, VF16=0...15

Table 13-42. VF Registers Mapping in the PF Space

Abbreviation	Virtual Address	Physical Address
VF_MBX_PSM_ARQH	0x00021080	0x0022E080 + 0x4*VF16, VF16=0...15
VF_MBX_PSM_ARQT	0x00021090	0x0022E0C0 + 0x4*VF16, VF16=0...15
VFQTX_COMM_DBLQ_DBELL	0x00022000 + 0x4*DBLQ, DBLQ=0...3	0x002D1400 + 0x4*DBLQ, DBLQ=0...255
VFINT_DYN_CTL	0x00023000 + 0x1000*n, n=0...7	0x00160000 + 0x4*INT, INT=0...2047
VFINT_ITR_0	0x00023004 + 0x1000*n, n=0...7	0x00154000 + 0x2000*n + 0x4*INT, n=0...2, INT=0...2047
VFINT_ITR_1	0x00023008 + 0x1000*n, n=0...7	0x00154000 + 0x2000*n + 0x4*INT, n=0...2, INT=0...2047
VFINT_ITR_2	0x0002300C + 0x1000*n, n=0...7	0x00154000 + 0x2000*n + 0x4*INT, n=0...2, INT=0...2047
VFQRX_TAIL	0x0002E000 + 0x4*QRX, QRX=0...255	0x00290000 + 0x4*QRX, QRX=0...2047
VFQTX_COMM_DBELL	0x00030000 + 0x4*DBQM, DBQM=0...255	0x002C0000 + 0x4*DBQM, DBQM=0...16383
MSIX_TADD	0x00000000 + 0x10*n, n=0...64	0x02E00000 + 0x10*n, n=0...2047
MSIX_TUADD	0x00000004 + 0x10*n, n=0...64	0x02E00004 + 0x10*n, n=0...2047
MSIX_TMSG	0x00000008 + 0x10*n, n=0...64	0x02E00008 + 0x10*n, n=0...2047
MSIX_TVCTRL	0x0000000C + 0x10*n, n=0...64	0x02E0000C + 0x10*n, n=0...2047
MSIX_PBA	0x00008000 + 0x4*n, n=0...2	0x02E08000 + 0x4*n, n=0...63

13.3.2 BAR0 Registers Summary

Table 13-43. VF - General Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00008800	VFGEN_RSTAT	VF Reset Status	13.3.3.1.1
0x0000E400	PFPCI_VF_FLUSH_DONE	VF Flush Done	13.3.3.1.2

Table 13-44. VF - Interrupt Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00002800 + 0x4*n + 0x40*m, n=0...15, m=0...2	VFINT_ITRN[n,m]	VF Interrupt Throttling N	13.3.3.2.1
0x00002C00 + 0x4*n + 0x100*m, n=0...63, m=0...2	VFINT_ITRN_64[n,m]	VF Interrupt Throttling N_64	13.3.3.2.2
0x00003800 + 0x4*n, n=0...63	VFINT_DYN_CTLN[n]	VF Interrupt Dynamic Control N	13.3.3.2.3
0x00004C00 + 0x4*n, n=0...2	VFINT_ITR0[n]	VF Interrupt Throttling Zero	13.3.3.2.4
0x00005C00	VFINT_DYN_CTL0	VF Interrupt Dynamic Control Zero	13.3.3.2.5

Table 13-45. VF - Control Queues Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00006000	VF_MBX_ARQBAH	VF Mailbox Receive Queue Base Address High	13.3.3.3.1
0x00006400	VF_MBX_ATQH	VF Mailbox Transmit Head	13.3.3.3.2
0x00006800	VF_MBX_ATQLEN	VF Mailbox Transmit Queue Length	13.3.3.3.3
0x00006C00	VF_MBX_ARQBAL	VF Mailbox Receive Queue Base Address Low	13.3.3.3.4
0x00007000	VF_MBX_ARQT	VF Mailbox Receive Tail	13.3.3.3.5
0x00007400	VF_MBX_ARQH	VF Mailbox Receive Head	13.3.3.3.6
0x00007800	VF_MBX_ATQBAH	VF Mailbox Transmit Queue Base Address High	13.3.3.3.7
0x00007C00	VF_MBX_ATQBAL	VF Mailbox Transmit Queue Base Address Low	13.3.3.3.8
0x00008000	VF_MBX_ARQLEN	VF Mailbox Receive Queue Length	13.3.3.3.9
0x00008400	VF_MBX_ATQT	VF Mailbox Transmit Tail	13.3.3.3.10

Table 13-46. VF - LAN Transmit and Receive Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00000000 + 0x4*DBQM, DBQM=0...255	QTX_TAIL[DBQM]	Transmit Queue Doorbell	13.3.3.4.1
0x00002000 + 0x4*QRX, QRX=0...255	QRX_TAIL[QRX]	Receive Queue Tail Update	13.3.3.4.2

Table 13-47. VF - Protocol Engine Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x00008C00	VFPE_IPCONFIG0	Protocol Engine VF IP Config 0	13.3.3.5.1
0x00009800	VFPE_CCQPHIGH	Protocol Engine VF Create CQP High	13.3.3.5.2
0x00009C00	VFPE_CQPERRCODES	Protocol Engine VF CQP Error Codes	13.3.3.5.3
0x0000A000	VFPE_CQPTAIL	Protocol Engine VF CQP Tail	13.3.3.5.4
0x0000A400	VFPE_AEQALLOC	Protocol Engine VF AEQ Allocate	13.3.3.5.5
0x0000A800	VFPE_TCPNOWTIMER	Protocol Engine VF TCP Now Timer	13.3.3.5.6
0x0000AC00	VFPE_CCQPLOW	Protocol Engine VF Create CQP Low	13.3.3.5.7
0x0000B000	VFPE_CQACK	Protocol Engine VF CQ Ack	13.3.3.5.8
0x0000B400	VFPE_CQARM	Protocol Engine VF CQ Arm	13.3.3.5.9
0x0000B800	VFPE_CCQPSTATUS	Protocol Engine VF Create CQP Status	13.3.3.5.10
0x0000BC00	VFPE_CQPDB	Protocol Engine VF CQP Doorbell	13.3.3.5.11
0x0000C000	VFPE_WQEALLOC	Protocol Engine VF WQE Allocate Register	13.3.3.5.12

Table 13-48. VF - Large VF Access Registers Summary

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0000F000	VF_MBX_CPM_ATQBAL	VF CPM Mailbox Transmit Queue Base Address Low	13.3.3.6.1
0x0000F010	VF_MBX_CPM_ATQBAH	VF CPM Mailbox Transmit Queue Base Address High	13.3.3.6.2
0x0000F020	VF_MBX_CPM_ATQLEN	VF CPM Mailbox Transmit Queue Length	13.3.3.6.3
0x0000F030	VF_MBX_CPM_ATQH	VF CPM Mailbox Transmit Head	13.3.3.6.4
0x0000F040	VF_MBX_CPM_ATQT	VF CPM Mailbox Transmit Tail	13.3.3.6.5
0x0000F050	VF_MBX_CPM_ARQBAL	VF CPM Mailbox Receive Queue Base Address Low	13.3.3.6.6
0x0000F060	VF_MBX_CPM_ARQBAH	VF CPM Mailbox Receive Queue Base Address High	13.3.3.6.7
0x0000F070	VF_MBX_CPM_ARQLEN	VF CPM Mailbox Receive Queue Length	13.3.3.6.8
0x0000F080	VF_MBX_CPM_ARQH	VF CPM Mailbox Receive Head	13.3.3.6.9
0x0000F090	VF_MBX_CPM_ARQT	VF CPM Mailbox Receive Tail	13.3.3.6.10
0x0000F100	VF_SB_CPM_ATQBAL	VF CPM Sideband Transmit Queue Base Address Low	13.3.3.6.11
0x0000F110	VF_SB_CPM_ATQBAH	VF CPM Sideband Transmit Queue Base Address High	13.3.3.6.12
0x0000F120	VF_SB_CPM_ATQLEN	VF CPM Sideband Transmit Queue Length	13.3.3.6.13
0x0000F130	VF_SB_CPM_ATQH	VF CPM Sideband Transmit Head	13.3.3.6.14
0x0000F140	VF_SB_CPM_ATQT	VF CPM Sideband Transmit Tail	13.3.3.6.15
0x0000F150	VF_SB_CPM_ARQBAL	VF CPM Sideband Receive Queue Base Address Low	13.3.3.6.16
0x0000F160	VF_SB_CPM_ARQBAH	VF CPM Sideband Receive Queue Base Address High	13.3.3.6.17

Table 13-48. VF - Large VF Access Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0000F170	VF_SB_CPM_ARQLEN	VF CPM Sideband Receive Queue Length	13.3.3.6.18
0x0000F180	VF_SB_CPM_ARQH	VF CPM Sideband Receive Head	13.3.3.6.19
0x0000F190	VF_SB_CPM_ARQT	VF CPM Sideband Receive Tail	13.3.3.6.20
0x00020000	VF_MBX_HLP_ATQBAL	VF HLP Mailbox Transmit Queue Base Address Low	13.3.3.6.21
0x00020010	VF_MBX_HLP_ATQBAH	VF HLP Mailbox Transmit Queue Base Address High	13.3.3.6.22
0x00020020	VF_MBX_HLP_ATQLEN	VF HLP Mailbox Transmit Queue Length	13.3.3.6.23
0x00020030	VF_MBX_HLP_ATQH	VF HLP Mailbox Transmit Head	13.3.3.6.24
0x00020040	VF_MBX_HLP_ATQT	VF HLP Mailbox Transmit Tail	13.3.3.6.25
0x00020050	VF_MBX_HLP_ARQBAL	VF HLP Mailbox Receive Queue Base Address Low	13.3.3.6.26
0x00020060	VF_MBX_HLP_ARQBAH	VF HLP Mailbox Receive Queue Base Address High	13.3.3.6.27
0x00020070	VF_MBX_HLP_ARQLEN	VF HLP Mailbox Receive Queue Length	13.3.3.6.28
0x00020080	VF_MBX_HLP_ARQH	VF HLP Mailbox Receive Head	13.3.3.6.29
0x00020090	VF_MBX_HLP_ARQT	VF HLP Mailbox Receive Tail	13.3.3.6.30
0x00021000	VF_MBX_PSM_ATQBAL	VF PSM Mailbox Transmit Queue Base Address Low	13.3.3.6.31
0x00021010	VF_MBX_PSM_ATQBAH	VF PSM Mailbox Transmit Queue Base Address High	13.3.3.6.32
0x00021020	VF_MBX_PSM_ATQLEN	VF PSM Mailbox Transmit Queue Length	13.3.3.6.33
0x00021030	VF_MBX_PSM_ATQH	VF PSM Mailbox Transmit Head	13.3.3.6.34
0x00021040	VF_MBX_PSM_ATQT	VF PSM Mailbox Transmit Tail	13.3.3.6.35
0x00021050	VF_MBX_PSM_ARQBAL	VF PSM Mailbox Receive Queue Base Address Low	13.3.3.6.36
0x00021060	VF_MBX_PSM_ARQBAH	VF PSM Mailbox Receive Queue Base Address High	13.3.3.6.37
0x00021070	VF_MBX_PSM_ARQLEN	VF PSM Mailbox Receive Queue Length	13.3.3.6.38
0x00021080	VF_MBX_PSM_ARQH	VF PSM Mailbox Receive Head	13.3.3.6.39
0x00021090	VF_MBX_PSM_ARQT	VF PSM Mailbox Receive Tail	13.3.3.6.40
0x00022000 + 0x4*DBLQ, DBLQ=0...3	VFQTX_COMM_DBLQ_DBELL[DBLQ]	Transmit Comm Scheduler Queue Doorbell	13.3.3.6.41
0x00023000 + 0x1000*n, n=0...7	VFINT_DYN_CTL[n]	VF Interrupt Dynamic Control	13.3.3.6.42
0x00023004 + 0x1000*n, n=0...7	VFINT_ITR_0[n]	VF Interrupt Throttling 0	13.3.3.6.43
0x00023008 + 0x1000*n, n=0...7	VFINT_ITR_1[n]	VF Interrupt Throttling 1	13.3.3.6.44
0x0002300C + 0x1000*n, n=0...7	VFINT_ITR_2[n]	VF Interrupt Throttling 2	13.3.3.6.45

Table 13-48. VF - Large VF Access Registers Summary [continued]

Offset/Alias Offset	Abbreviation	Name	Section Reference
0x0002E000 + 0x4*QRX, QRX=0...255	VFQRX_TAIL[QRX]	Global Receive Queue Tail	13.3.3.6.46
0x00030000 + 0x4*DBQM, DBQM=0...255	VFQTX_COMM_DBELL[DBQM]	Transmit Comm Scheduler Queue Doorbell	13.3.3.6.47

13.3.3 Detailed Register Descriptions - VF BAR0

13.3.3.1 VF - General Registers

This section describes the registers allocated to a VF for generic control and status. These registers are tied to the VF and are not dependent of any resource allocation.

13.3.3.1.1 VF Reset Status - VFGEN_RSTAT (0x00008800; RW)

Field definitions are the same as those defined in Section 13.2.2.1.1.

13.3.3.1.2 VF Flush Done - PFPCI_VF_FLUSH_DONE (0x0000E400; RO)

Field definitions are the same as those defined in Section 13.2.2.3.24.

13.3.3.2 VF - Interrupt Registers

13.3.3.2.1 VF Interrupt Throttling N - VFINT_ITRN[n,m] (0x00002800 + 0x4*n + 0x40*m, n=0...15, m=0...2; RW)

Field definitions are the same as those defined in Section 13.2.2.15.7.

13.3.3.2.2 VF Interrupt Throttling N_64 - VFINT_ITRN_64[n,m] (0x00002C00 + 0x4*n + 0x100*m, n=0...63, m=0...2; RW)

Field definitions are the same as those defined in Section 13.2.2.15.7.

13.3.3.2.3 VF Interrupt Dynamic Control N - VFINT_DYN_CTLN[n] (0x00003800 + 0x4*n, n=0...63; RW)

Field definitions are the same as those defined in Section 13.2.2.15.10.

13.3.3.2.4 VF Interrupt Throttling Zero - VFINT_ITR0[n] (0x00004C00 + 0x4*n, n=0...2; RW)

Field definitions are the same as those defined in Section 13.2.2.15.7.

13.3.3.2.5 VF Interrupt Dynamic Control Zero - VFINT_DYN_CTL0 (0x00005C00; RW)

Field definitions are the same as those defined in Section 13.2.2.15.10.

13.3.3.3 VF - Control Queues Registers

VF exposed control queues.

13.3.3.3.1 VF Mailbox Receive Queue Base Address High - VF_MBX_ARQBAH (0x00006000; RW)

Field definitions are the same as those defined in Section 13.2.2.22.61.

13.3.3.3.2 VF Mailbox Transmit Head - VF_MBX_ATQH (0x00006400; RW)

Field definitions are the same as those defined in Section 13.2.2.22.58.

13.3.3.3.3 VF Mailbox Transmit Queue Length - VF_MBX_ATQLEN (0x00006800; RW)

Field definitions are the same as those defined in Section 13.2.2.22.57.

13.3.3.3.4 VF Mailbox Receive Queue Base Address Low - VF_MBX_ARQBAL (0x00006C00; RW)

Field definitions are the same as those defined in Section 13.2.2.22.60.

13.3.3.3.5 VF Mailbox Receive Tail - VF_MBX_ARQT (0x00007000; RW)

Field definitions are the same as those defined in Section 13.2.2.22.64.

13.3.3.3.6 VF Mailbox Receive Head - VF_MBX_ARQH (0x00007400; RW)

Field definitions are the same as those defined in Section 13.2.2.22.63.

13.3.3.3.7 VF Mailbox Transmit Queue Base Address High - VF_MBX_ATQBAH (0x00007800; RW)

Field definitions are the same as those defined in Section 13.2.2.22.56.

13.3.3.3.8 VF Mailbox Transmit Queue Base Address Low - VF_MBX_ATQBAL (0x00007C00; RW)

Field definitions are the same as those defined in Section 13.2.2.22.55.

13.3.3.3.9 VF Mailbox Receive Queue Length - VF_MBX_ARQLEN (0x00008000; RW)

Field definitions are the same as those defined in Section 13.2.2.22.62.

13.3.3.3.10 VF Mailbox Transmit Tail - VF_MBX_ATQT (0x00008400; RW)

Field definitions are the same as those defined in Section 13.2.2.22.59.

13.3.3.4 VF LAN Transmit and Receive Registers

13.3.3.4.1 Transmit Queue Doorbell - QTX_TAIL[DBQM] (0x00000000 + 0x4*DBQM, DBQM=0...255; RW)

Field definitions are the same as those defined in Section 13.2.2.25.27.

13.3.3.4.2 Receive Queue Tail Update - QRX_TAIL[QRX] (0x00002000 + 0x4*QRX, QRX=0...255; RW)

Field definitions are the same as those defined in Section 13.2.2.26.22.

13.3.3.5 VF - Protocol Engine Registers

13.3.3.5.1 Protocol Engine VF IP Config 0 - VFPE_IPCONFIG0 (0x00008C00; RW)

Field definitions are the same as those defined in Section 13.2.2.28.23.

13.3.3.5.2 Protocol Engine VF Create CQP High - VFPE_CCQPHIGH (0x00009800; RW)

Field definitions are the same as those defined in Section 13.2.2.28.22.

13.3.3.5.3 Protocol Engine VF CQP Error Codes - VFPE_CQPERRCODES (0x00009C00; RO)

Field definitions are the same as those defined in Section 13.2.2.28.24.

13.3.3.5.4 Protocol Engine VF CQP Tail - VFPE_CQPTAIL (0x0000A000; RO)

Field definitions are the same as those defined in Section 13.2.2.28.2.

13.3.3.5.5 Protocol Engine VF AEQ Allocate - VFPE_AEQALLOC (0x0000A400; RW)

Field definitions are the same as those defined in Section 13.2.2.28.7.

13.3.3.5.6 Protocol Engine VF TCP Now Timer - VFPE_TCPNOWTIMER (0x0000A800; RO)

Field definitions are the same as those defined in Section 13.2.2.28.25.

13.3.3.5.7 Protocol Engine VF Create CQP Low - VFPE_CCQPLOW (0x0000AC00; RW)

Field definitions are the same as those defined in Section 13.2.2.28.21.

13.3.3.5.8 Protocol Engine VF CQ Ack - VFPE_CQACK (0x0000B000; RW)

Field definitions are the same as those defined in Section 13.2.2.28.6.

13.3.3.5.9 Protocol Engine VF CQ Arm - VFPE_CQARM (0x0000B400; RW)

Field definitions are the same as those defined in Section 13.2.2.28.5.

13.3.3.5.10 Protocol Engine VF Create CQP Status - VFPE_CCQPSTATUS (0x0000B800; RO)

Field definitions are the same as those defined in Section 13.2.2.28.20.

13.3.3.5.11 Protocol Engine VF CQP Doorbell - VFPE_CQPDB (0x0000BC00; RW)

Field definitions are the same as those defined in Section 13.2.2.28.1.

13.3.3.5.12 Protocol Engine VF WQE Allocate Register - VFPE_WQEALLOC (0x0000C000; RW)

Field definitions are the same as those defined in Section 13.2.2.28.18.

13.3.3.6 VF - Large VF Access Registers

13.3.3.6.1 VF CPM Mailbox Transmit Queue Base Address Low - VF_MBX_CPM_ATQBAL (0x0000F000; RW)

Field definitions are the same as those defined in Section 13.2.2.22.65.

13.3.3.6.2 VF CPM Mailbox Transmit Queue Base Address High - VF_MBX_CPM_ATQBAH (0x0000F010; RW)

Field definitions are the same as those defined in Section 13.2.2.22.66.

13.3.3.6.3 VF CPM Mailbox Transmit Queue Length - VF_MBX_CPM_ATQLEN (0x0000F020; RW)

Field definitions are the same as those defined in Section 13.2.2.22.67.

13.3.3.6.4 VF CPM Mailbox Transmit Head - VF_MBX_CPM_ATQH (0x0000F030; RW)

Field definitions are the same as those defined in Section 13.2.2.22.68.

13.3.3.6.5 VF CPM Mailbox Transmit Tail - VF_MBX_CPM_ATQT (0x0000F040; RW)

Field definitions are the same as those defined in Section 13.2.2.22.69.

13.3.3.6.6 VF CPM Mailbox Receive Queue Base Address Low - VF_MBX_CPM_ARQBAL (0x0000F050; RW)

Field definitions are the same as those defined in Section 13.2.2.22.70.

13.3.3.6.7 VF CPM Mailbox Receive Queue Base Address High - VF_MBX_CPM_ARQBAH (0x0000F060; RW)

Field definitions are the same as those defined in Section 13.2.2.22.71.

13.3.3.6.8 VF CPM Mailbox Receive Queue Length - VF_MBX_CPM_ARQLEN (0x0000F070; RW)

Field definitions are the same as those defined in Section 13.2.2.22.72.

13.3.3.6.9 VF CPM Mailbox Receive Head - VF_MBX_CPM_ARQH (0x0000F080; RW)

Field definitions are the same as those defined in Section 13.2.2.22.73.

13.3.3.6.10 VF CPM Mailbox Receive Tail - VF_MBX_CPM_ARQT (0x0000F090; RW)

Field definitions are the same as those defined in Section 13.2.2.22.74.

13.3.3.6.11 VF CPM Sideband Transmit Queue Base Address Low - VF_SB_CPM_ATQBAL (0x0000F100; RW)

Field definitions are the same as those defined in Section 13.2.2.22.145.

13.3.3.6.12 VF CPM Sideband Transmit Queue Base Address High - VF_SB_CPM_ATQBAH (0x0000F110; RW)

Field definitions are the same as those defined in Section 13.2.2.22.146.

13.3.3.6.13 VF CPM Sideband Transmit Queue Length - VF_SB_CPM_ATQLEN (0x0000F120; RW)

Field definitions are the same as those defined in Section 13.2.2.22.147.

13.3.3.6.14 VF CPM Sideband Transmit Head - VF_SB_CPM_ATQH (0x0000F130; RW)

Field definitions are the same as those defined in Section 13.2.2.22.148.

13.3.3.6.15 VF CPM Sideband Transmit Tail - VF_SB_CPM_ATQT (0x0000F140; RW)

Field definitions are the same as those defined in Section 13.2.2.22.149.

13.3.3.6.16 VF CPM Sideband Receive Queue Base Address Low - VF_SB_CPM_ARQBAL (0x0000F150; RW)

Field definitions are the same as those defined in Section 13.2.2.22.150.

13.3.3.6.17 VF CPM Sideband Receive Queue Base Address High - VF_SB_CPM_ARQBAH (0x0000F160; RW)

Field definitions are the same as those defined in Section 13.2.2.22.151.

13.3.3.6.18 VF CPM Sideband Receive Queue Length - VF_SB_CPM_ARQLEN (0x0000F170; RW)

Field definitions are the same as those defined in Section 13.2.2.22.152.

13.3.3.6.19 VF CPM Sideband Receive Head - VF_SB_CPM_ARQH (0x0000F180; RW)

Field definitions are the same as those defined in Section 13.2.2.22.153.

13.3.3.6.20 VF CPM Sideband Receive Tail - VF_SB_CPM_ARQT (0x0000F190; RW)

Field definitions are the same as those defined in Section 13.2.2.22.154.

13.3.3.6.21 VF HLP Mailbox Transmit Queue Base Address Low - VF_MBX_HLP_ATQBAL (0x00020000; RW)

Field definitions are the same as those defined in Section 13.2.2.22.75.

13.3.3.6.22 VF HLP Mailbox Transmit Queue Base Address High - VF_MBX_HLP_ATQBAH (0x00020010; RW)

Field definitions are the same as those defined in Section 13.2.2.22.76.

13.3.3.6.23 VF HLP Mailbox Transmit Queue Length - VF_MBX_HLP_ATQLEN (0x00020020; RW)

Field definitions are the same as those defined in Section 13.2.2.22.77.

13.3.3.6.24 VF HLP Mailbox Transmit Head - VF_MBX_HLP_ATQH (0x00020030; RW)

Field definitions are the same as those defined in Section 13.2.2.22.78.

13.3.3.6.25 VF HLP Mailbox Transmit Tail - VF_MBX_HLP_ATQT (0x00020040; RW)

Field definitions are the same as those defined in Section 13.2.2.22.79.

13.3.3.6.26 VF HLP Mailbox Receive Queue Base Address Low - VF_MBX_HLP_ARQBAL (0x00020050; RW)

Field definitions are the same as those defined in Section 13.2.2.22.80.

13.3.3.6.27 VF HLP Mailbox Receive Queue Base Address High - VF_MBX_HLP_ARQBAH (0x00020060; RW)

Field definitions are the same as those defined in Section 13.2.2.22.81.

13.3.3.6.28 VF HLP Mailbox Receive Queue Length - VF_MBX_HLP_ARQLEN (0x00020070; RW)

Field definitions are the same as those defined in Section 13.2.2.22.82.

13.3.3.6.29 VF HLP Mailbox Receive Head - VF_MBX_HLP_ARQH (0x00020080; RW)

Field definitions are the same as those defined in Section 13.2.2.22.83.

13.3.3.6.30 VF HLP Mailbox Receive Tail - VF_MBX_HLP_ARQT (0x00020090; RW)

Field definitions are the same as those defined in Section 13.2.2.22.84.

13.3.3.6.31 VF PSM Mailbox Transmit Queue Base Address Low - VF_MBX_PSM_ATQBAL (0x00021000; RW)

Field definitions are the same as those defined in Section 13.2.2.22.85.

13.3.3.6.32 VF PSM Mailbox Transmit Queue Base Address High - VF_MBX_PSM_ATQBAH (0x00021010; RW)

Field definitions are the same as those defined in Section 13.2.2.22.86.

13.3.3.6.33 VF PSM Mailbox Transmit Queue Length - VF_MBX_PSM_ATQLEN (0x00021020; RW)

Field definitions are the same as those defined in Section 13.2.2.22.87.

13.3.3.6.34 VF PSM Mailbox Transmit Head - VF_MBX_PSM_ATQH (0x00021030; RW)

Field definitions are the same as those defined in Section 13.2.2.22.88.

13.3.3.6.35 VF PSM Mailbox Transmit Tail - VF_MBX_PSM_ATQT (0x00021040; RW)

Field definitions are the same as those defined in Section 13.2.2.22.89.

13.3.3.6.36 VF PSM Mailbox Receive Queue Base Address Low - VF_MBX_PSM_ARQBAL (0x00021050; RW)

Field definitions are the same as those defined in Section 13.2.2.22.90.

13.3.3.6.37 VF PSM Mailbox Receive Queue Base Address High - VF_MBX_PSM_ARQBAH (0x00021060; RW)

Field definitions are the same as those defined in Section 13.2.2.22.91.

13.3.3.6.38 VF PSM Mailbox Receive Queue Length - VF_MBX_PSM_ARQLEN (0x00021070; RW)

Field definitions are the same as those defined in Section 13.2.2.22.92.

13.3.3.6.39 VF PSM Mailbox Receive Head - VF_MBX_PSM_ARQH (0x00021080; RW)

Field definitions are the same as those defined in Section 13.2.2.22.93.

13.3.3.6.40 VF PSM Mailbox Receive Tail - VF_MBX_PSM_ARQT (0x00021090; RW)

Field definitions are the same as those defined in Section 13.2.2.22.94.

13.3.3.6.41 Transmit Comm Scheduler Queue Doorbell - VFQTX_COMM_DBLQ_DBELL[DBLQ] (0x00022000 + 0x4*DBLQ, DBLQ=0...3; RW)

Field definitions are the same as those defined in Section 13.2.2.25.29.

13.3.3.6.42 VF Interrupt Dynamic Control - VFINT_DYN_CTL[n] (0x00023000 + 0x1000*n, n=0...7; RW)

Field definitions are the same as those defined in Section 13.2.2.15.10.

13.3.3.6.43 VF Interrupt Throttling 0 - VFINT_ITR_0[n] (0x00023004 + 0x1000*n, n=0...7; RW)

Field definitions are the same as those defined in Section 13.2.2.15.7.

13.3.3.6.44 VF Interrupt Throttling 1 - VFINT_ITR_1[n] (0x00023008 + 0x1000*n, n=0...7; RW)

Field definitions are the same as those defined in Section 13.2.2.15.7.

13.3.3.6.45 VF Interrupt Throttling 2 - VFINT_ITR_2[n] (0x0002300C + 0x1000*n, n=0...7; RW)

Field definitions are the same as those defined in Section 13.2.2.15.7.

13.3.3.6.46 Global Receive Queue Tail - VFQRX_TAIL[QRX] (0x0002E000 + 0x4*QRX, QRX=0...255; RW)

Field definitions are the same as those defined in Section 13.2.2.26.22.

13.3.3.6.47 Transmit Comm Scheduler Queue Doorbell - VFQTX_COMM_DBELL[DBQM] (0x00030000 + 0x4*DBQM, DBQM=0...255; RW)

Field definitions are the same as those defined in Section 13.2.2.25.27.

13.3.4 BAR3 Registers Summary

Table 13-49. VF - MSI-X Table Registers Summary

Offset / Alias Offset	Abbreviation	Name	Section Reference
0x00000000 + 0x10*n, n=0...2047	MSIX_TADD[n]	MSI-X Message Address Low	Section 13.3.5.1.1
0x00000004 + 0x10*n, n=0...2047	MSIX_TUADD[n]	MSI-X Message Address High	Section 13.3.5.1.2
0x00000008 + 0x10*n, n=0...2047	MSIX_TMSG[n]	MSI-X Message Data	Section 13.3.5.1.3
0x0000000C + 0x10*n, n=0...2047	MSIX_TVCTRL[n]	MSI-X Vector Control	Section 13.3.5.1.4
0x00008000 + 0x4*n, n=0...63	MSIX_PBA[n]	MSI-X PBA Structure	Section 13.3.5.1.5

13.3.5 Detailed Register Descriptions - VF BAR3

13.3.5.1 VF - MSI-X Table Registers

13.3.5.1.1 MSI-X Message Address Low - MSIX_TADD[n] (0x00000000 + 0x10*n, n=0...64; RW)

Field definitions are the same as those defined in Section 13.2.4.1.1.

13.3.5.1.2 MSI-X Message Address High - MSIX_TUADD[n] (0x00000004 + 0x10*n, n=0...64; RW)

Field definitions are the same as those defined in Section 13.2.4.1.2.

13.3.5.1.3 MSI-X Message Data - MSIX_TMSG[n] (0x00000008 + 0x10*n, n=0...64; RW)

Field definitions are the same as those defined in Section 13.2.4.1.3.

13.3.5.1.4 MSI-X Vector Control - MSIX_TVCTRL[n] (0x0000000C + 0x10*n, n=0...64; RW)

Field definitions are the same as those defined in Section 13.2.4.1.4.

13.3.5.1.5 MSI-X PBA Structure - MSIX_PBA[n] (0x00008000 + 0x4*n, n=0...2; RO)

Field definitions are the same as those defined in Section 13.2.4.1.5.

NOTE: ***This page intentionally left blank.***

Chapter 14 PCIe Programming Interface

14.1 Overview

The E810 supports the following configuration register sets:

- PCI basic configuration registers (see Section 14.2).
- PCI and PCIe capabilities in the PCI configuration space (see Section 14.3).
  - Includes the PCIe capability structure (see Section 14.3.5).
- PCIe capabilities residing in the PCIe extended configuration Space (see Section 14.4).
- SR-IOV VF configuration space (see Section 14.5).

14.1.1 Functions Mapping

The E810 is a multi-function device with the following characteristics:

- Up to eight8 physical functions (PFs)
- Up to 256 SR-IOV Virtual Functions (VFs)
  - Each PF can be allocated a different number of VFs in the range {0,...,256-1} as long as the total number of VFs does not exceed 256.

The following rule applies regarding allocation of PCI functions to LAN ports:

- Each PCI function is associated with a single LAN port as indicated in the PFGEN_PORTNUM.*PORT_NUM* register field.

Note: NVM programming notes:

Set PFGEN_PORTNUM_CAR in NVM POR auto-load section.

Set PFGEN_PORTNUM in NVM CORER auto-load section.

The number of enabled physical functions might be deducted from the PFPCI_STATUS1.*FUNC_VALID* bits (one per PF). See Section 4.5.4 on how enabled functions are determined.

The ARI capability enables interpretation of the device number part of the RID as part of the function number inside a device. Thus, a single device can span more than eight physical or virtual functions. Table 14-1 and Table 14-2 map the physical functions to PCI Requester ID.

Table 14-1. RID per PF - ARI Mode

PF#	B,D,F	Binary	Notes
PF 0	B,0,0	B,00000,000	PF #0
PF 1	B,0,1	B,00000,001	PF #1
PF 2	B,0,2	B,00000,010	PF #2
PF 7	B,0,7	B,00000,111	PF #7

Table 14-2. RID per PF - non-ARI Mode

PF#	B,D,F	Binary	Notes
PF 0	B,0,0	B,00000,000	PF #0
PF 1	B,0,1	B,00000,001	PF #1
PF 2	B,0,2	B,00000,010	PF #2
PF 7	B,0,7	B,00000,111	PF #7

The Requester ID of a PF (bus, device, and function numbers) is captured in the PF_FUNC_RID register.

14.1.1.1 Support for Dynamic Changes

The E810 captures the bus number and device number per each configuration write request. However, a dynamic change of the 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.

14.1.2 Supported Features

Table 14-3 lists the PCI and PCIe capabilities supported per PCI function type. Some capabilities do not necessarily appear with each function or in all cases. See Section 14.3 and Section 14.4 for details on specific capabilities.

Table 14-3. PCI Capabilities Supported by Function

PCI Capability	PFs	VFs	Section Reference
PCI Configuration	Mandatory	Mandatory	14.2
Power Management	Yes	Yes	14.3.1
MSI	Yes	No	14.3.2
MSI-X	Yes	Yes	14.3.3
Vital Product Data (VPD)	Yes	No	14.3.4
PCIe	Yes	Yes	14.3.5
Advanced Error Reporting (AER)	Yes	Yes	14.4.1
Device Serial Number	Yes	No (N/A)	14.4.2
Alternative RID Interpretation (ARI)	Yes	Yes	14.4.3
Single Root I/O Virtualization (SR-IOV)	Yes	No (N/A)	14.4.4
TPH Requester	Yes	Yes	14.4.5
Access Control Services (ACS)	Yes	Yes	14.4.6
Secondary PCIe	Yes	No (N/A)	14.4.7
Data Link Feature	Yes	No (N/A)	14.4.8
Process ID (PASID)	Yes	No (N/A)	14.4.9
Physical Layer 16.0 GT/s Extended Capability	Yes	No (N/A)	14.4.10
Lane Margining at the Receiver Capability	Yes	No (N/A)	14.4.11

14.2 PCI Configuration Space

Configuration registers are assigned one of the attributes listed in Table 14-4.

14.2.1 Register Attributes

Table 14-4 lists the register attributes used in this section.

Table 14-4. 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 neither initialized nor modified by PCIe in-band reset or FLR. Specific bits listed below are also not reset on PERST# when aux power consumption is enabled.
RWS	**Read/Write register with Sticky bits** Register bits are read/write and can be either set or cleared by software to the desired state. Bits are neither initialized nor modified by PCIe in-band reset or FLR. Specific bits listed below are also not reset on PERST# when aux power consumption 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 neither initialized nor modified by PCIe in-band reset or FLR. Specific bits listed in the sections that follow are also not reset on PERST# when aux power consumption 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.

14.2.2 Reset Rules

Reset of the PCI configuration space (including any capability lists) is per the PCIe specification. Several cases require special attention.

14.2.2.1 Sticky Registers

The following sticky register fields are also not reset on PERST# when aux power consumption is enabled (AUX_PWR pin is set).

- PME-related fields:
  - Power Management Capabilities register — *PME_En* bit.
  - Power Management Capabilities register — *PME_Status* bit.
  - Device Control register — *Aux Power PM Enable* bit.
  - The function Requester ID.
- AER-related fields:
  - Uncorrectable Error Status register.
  - Uncorrectable Error Mask register.
  - Uncorrectable Error Severity register.
  - Correctable Error Status register.
  - Correctable Error Mask register.
  - Advanced Error Capabilities and Control register.
  - Header log.
- Physical Layer 16.0 GT/s related fields:
  - Physical Layer 16.0 GT/s Status Register
  - 16.0 GT/s Local Data Parity Mismatch Status Register
  - 16.0 GT/s First Re-timer Data Parity Mismatch Status Register
  - 16.0 GT/s Second Re-timer Data Parity Mismatch Status Register

14.2.2.2 Reset on FLR

The following registers are not affected by FLR:

- The *Max Payload Size* field in the Device Control register.
- The *Active State Link PM Control* field in the Link Control register.
- The *Common Clock Configuration* field in the Link Control register.
- The *Extended Sync* field in the Link Control register.
- The *Link Equalization Request* field in the Link Status 2 register.

14.2.3 PCI Configuration Space Summary

Table 14-5 lists the PCI configuration registers, while their detailed description is given in the sections that follow. Fields that have meaningful default values are indicated in parenthesis — (**value**).

The PCI configuration space from address 0x40 on is allocated for PCI capability structures as described in Section 14.3. However, the region of 0x98-0x9F (8 bytes) is dedicated to an address/data port, which provides access to the device I/O address space (see Section 13.1.1.6 for more details).

Table 14-5. PCI Configuration Space - PF

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 (0x10)
	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)

14.2.4 Sharing Among PCI Functions

The E810 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. This section summarizes configuration sharing of the fixed PCI configuration space. See the description of each PCI capability structure for configuration sharing within it. Also, the description of each field describes special considerations regarding configuration sharing.

Table 14-6. Configuration Sharing of PCI Configuration Space

Field	Sub-Field	Shared?	Replicated?	Comments	Section Reference
Vendor ID	Vendor ID	X			14.2.5.1
Device ID	Device ID		X		14.2.5.2
Command Register	I/O Access Enable		X	Issue UR per PF if disabled.	14.2.5.3
	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.	
Status Register	Interrupt Status		X		14.2.5.4
	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			14.2.5.5
Class Code Register			X	Per function type.	14.2.5.6
Cache Line Size Register			X	Does not affect device behavior.	14.2.5.7
Latency Timer		X		Hard-wired to 0x00 in PCIe.	14.2.5.8
Header Type Register		X			14.2.5.9
BARs	Memory BAR		X		14.2.6
	I/O BAR		X		
	MSI-X BAR		X	See MSI-X capability.	
I/O Bar Mapping	IOADDR, IODATA		X		14.2.6.1
Subsystem Vendor ID		X			14.2.5.10
Subsystem ID			X		14.2.5.11
Expansion ROM		X		Each PF has its own BAR.	14.2.6.2
Cap_Ptr Register		X			14.2.5.12
Interrupt Line Register			X	Just store the register value.	14.2.5.13
Interrupt Pin Register			X	Separate interrupt number (A-D) per PF.	14.2.5.14
Min Grant		X			14.2.5.15
Max Latency		X			

14.2.5 Mandatory PCI Configuration Registers - Except BARs

14.2.5.1 Vendor ID Register (0x0; RO)

This is a read-only register that has the same value for all PCI functions.

- Vendor ID is loaded from the NVM if the GLPCI_CAPSUP.*LOAD_DEV_ID* bit is set.
- The value for all PFs is loaded from the NVM GLPCI_VENDORID.*VENDOR_ID* field.

14.2.5.2 Device ID Register (0x2; RO)

This is a read-only register that identifies individual E810 PCI functions. All functions have the same default value of 0x1590 for the E810-CAM2/CAM1 (25x25 mm package) and 0x1598 for the E810-XXVAM2 (21x21 mm package), and can be auto-loaded from the NVM during initialization with different values for each function as well as the dummy function (see Section 4.5.4.1 for dummy function details).

Device ID is loaded from the NVM according to the following rules:

- The Device ID is loaded from the NVM if the GLPCI_CAPSUP.*LOAD_DEV_ID* bit is set.
- The value of each PF is loaded to the respective PFPCI_DEVID.*PF_DEV_ID* field.

Table 14-7 describes the values set in this register from NVM according to the different PHY connections.

Table 14-7. Device IDs and Branding Strings

Device ID	Interface	Branding String
0x1590	Default Device ID (E810-CAM2/CAM1)	Intel® Ethernet Connection E810-C
0x1598	Default Device ID (E810-XXVAM2)	Intel® Ethernet Connection E810-XXV
0x1889	Default Virtual function device ID	Intel® Ethernet Adaptive Virtual Function
0x1591	backplane (E810-CAM2/CAM1)	Intel® Ethernet Controller E810-C for backplane
0x1599	backplane (E810-XXVAM2)	Intel® Ethernet Controller E810-XXV for backplane
0x1592	QSFP (E810-CAM2/CAM1)	Intel® Ethernet Controller E810-C for QSFP
0x159A	QSFP (E810-XXVAM2)	Intel® Ethernet Controller E810-XXV for QSFP
0x1593	SFP (E810-CAM2/CAM1)	Intel® Ethernet Controller E810-C for SFP
0x159B	SFP (E810-XXVAM2)	Intel® Ethernet Controller E810-XXV for SFP
0x1594	10GBASE-T (E810-CAM2/CAM1)	Intel® Ethernet Controller E810-C/X557-AT 10GBASE-T
0x159C	10GBASE-T (E810-XXVAM2)	Intel® Ethernet Controller E810-XXV/X557-AT 10GBASE-T
0x1595	1GBASE-T (E810-CAM2/CAM1)	Intel® Ethernet Controller E810-C 1GbE
0x159D	1GBASE-T (E810-XXVAM2)	Intel® Ethernet Controller E810-XXV 1GbE
0x1596-7	Reserved for future use (E810-CAM2/CAM1)	
0x159E-F	Reserved for future use (E810-XXVAM2)	

14.2.5.3 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 (see Table 14-6).

Bits	Init.	Type	Description
0	0b	RW (see comment)	**I/O Access Enable** This bit is RO if an I/O BAR is not supported by the device.
1	0b	RW	**Memory Access Enable**
2	0b	RW	**Enable Mastering** Also named Bus Master Enable (BME). • LAN functions RW field. • Dummy function RO as zero field.
3	0b	RO	**Special Cycle Monitoring** Hard-wired to 0b.
4	0b	RO	**MWI Enable** Hard-wired to 0b.
5	0b	RO	**Palette Snoop Enable** Hard-wired to 0b.
6	0b	RW	**Parity Error Response**
7	0b	RO	**Wait Cycle Enable** Hard-wired to 0b.
8	0b	RW	**SERR# Enable**
9	0b	RO	**Fast Back-to-Back Enable** Hard-wired to 0b.
10	0b	RW	**Interrupt Disable** When set, devices are prevented from generating legacy interrupt messages. RO as 0b for a dummy function.
15:11	0x0	RO	Reserved.

14.2.5.4 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.

Bits	Init.	Type	Description
0	0b	RO	**Immediate Readiness** Not supported in the E810.
2:1	00b		Reserved.
3	0b	RO	**Interrupt Status**[1]
4	1b	RO	**New Capabilities** Indicates that a device implements extended capabilities. The E810 sets this bit and implements a capabilities list to indicate that it supports PCI and PCIe capabilities.
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**

Bits	Init.	Type	Description
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.

14.2.5.5 Revision Register (0x8; RO)

The default revision ID of this device is 0x00 for A0 step. The default value is readable through the GLPCI_DREVID register. The value in the Revision register is a logic XOR between the default value and a value loaded from the NVM, reflected via the GLPCI_REVID register.

The default Revision ID for each E810 SKU is:

- E810-C in 25x25 mm package, B0 Step: 0x01
- E810-XXV in 21x21 mm package, A0 Step: 0x01

14.2.5.6 Class Code Register (0x9; RO)

The class code is a read-only value that identifies the device functionality:

- 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.

The device default value is the class code of an Ethernet adapter. The value is overwritten from the NVM. It is loaded to the RO PFPCI_CLASS register.

14.2.5.7 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. All functions are initialized to the same value of 0x00 (specification required).

14.2.5.8 Latency Timer (0xD; RO)

Not used. Hard-wired to 0b.

14.2.5.9 Header Type Register (0xE; RO)

This indicates if a device is single- or multi-function:

- **0x00** — A single function is enabled.
- **0x80** — At least two functions are enabled.

Notes:

- Dummy functions are considered as regular functions in this regard.
- SR-IOV VFs are not counted in the setting of the Header Type register.
- Functions might be disabled during the power on reset flow (through strapping pins, SMASH/CLP commands, NC-SI commands) affecting this bit. See Section 4.5.4.

14.2.5.10 Subsystem Vendor ID Register (0x2C; RO)

This value is loaded from the NVM if the GLPCI_CAPSUP.*LOAD_SUBSYS_ID* bit is set. A value of 0x8086 is the default for this field. It can be read through the GLPCI_SUBVENID register.

14.2.5.11 Subsystem ID Register (0x2E; RO)

This value is loaded from the NVM if the GLPCI_CAPSUP.*LOAD_SUBSYS_ID* bit is set. Each PF is loaded to the respective PFPCI_SUBSYSID.*PF_SUBSYS_ID* field.

14.2.5.12 Capabilities Pointer Register (0x34; RO)

The *Capabilities Pointer* field (Cap_Ptr) is an 8-bit field that provides an offset in the E810's PCI configuration space for the location of the first item in the capabilities linked list. The E810 supports this field and implements a capabilities list. Its value is 0x40, which is the address of the first entry: PCI power management.

14.2.5.13 Interrupt Line Register (0x3C; RW)

Read/write register programmed by software to indicate which of the system interrupt request lines the E810's interrupt pin is bound to. Refer to the PCI definition for more details.

14.2.5.14 Interrupt Pin Register (0x3D; RO)

Read-only register. A value of 0x1...0x4 indicates that this function implements a legacy interrupt on INTA#...INTD# respectively. Loaded from the NVM. It can be read through the RO PFPCI_CNF register. Device default value is 0x0.

If only a single function is enabled, the *Interrupt Pin* field of the enabled function reports INTA# usage.

Reports a value of 0x0 (function uses no legacy interrupt message) for a dummy function.

14.2.5.15 MIN_GNT and MAX_LAT (0x3E; RO)

Not used. Hard-wired to 0b.

14.2.6 Mandatory PCI Configuration Registers - BARs

14.2.6.1 Memory and I/O BARs (0x10 - 0x27; RW)

BARs are used to map E810 register space of the device functions. The E810 has a memory BAR, an I/O BAR (optional) an MSI-X BAR as listed in Table 14-8. The BARs location and sizes are listed in Table 14-9. The fields within each BAR are then listed in Table 14-10.

Table 14-8. E810 BAR Description

Mapping Windows	Mapping Description
Memory BAR	The internal registers memories, Protocol Engine doorbells, and external Flash devices are accessed as direct memory-mapped offsets from the BAR. Software can access a DWord or 64 bits.
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 NVM configuration. Device default value is that an I/O BAR is not supported. The state of the I/O BAR is exposed in the PFPCI_CNF register. This BAR is not present in dummy functions.
MSI-X BAR	The MSI-X vectors and PBA structures are accessed as direct memory-mapped offsets from the MSI-X BAR. Software can access DWord entities. This BAR is not present in dummy functions.

Table 14-9. E810 Base Address Setting in 64-bit BARs Mode

BAR	Addr	31 5	4	3	2	1	0
0	0x10	Memory CSR + FLASH BAR Low: 31:24 = RW 23:18 = RW or 0b 17:4 = 0b		0/1	1	0	0
1	0x14	Memory CSR + FLASH BAR High (RW)					
2	0x18	IO BAR (RW — 31:5) (optional)	0	0	0	0	1
3	0x1C	MSI-X BAR Low (RW — 31:15; RO 0b — 14:4)		0/1	1	0	0
4	0x20	MSI-X BAR High (RW)					
5	0x24	Reserved (RO — 0)					

Table 14-10. Base Address Registers Fields

Field	Bits	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** This field is loaded from the NVM. Hardware default is 64-bit.
Prefetch Memory	3	RO	**Prefetch Memory** 0b = Non-prefetchable space. 1b = Prefetchable space. This bit is loaded from NVM. This bit should be set only on systems that do not generate prefetchable cycles. Device default is 1b (prefetchable). It is exposed in the GLPCI_LBARCTRL register.

Table 14-10. Base Address Registers Fields [continued]

Field	Bits	Type	Description	
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 sizes. Initial value of the RW fields is 0x0.	
			Mapping Window	RO bits
			Memory space for CSRs, PE doorbells, and Flash memory access and 4K pages.	16:4 for 128 KB 17:4 for 256 KB and so on...
			MSI-X space is 32 KB.	14:4
			I/O space size is 32 bytes.	4:0

14.2.6.2 Expansion ROM Base Address Register (0x30; RW)

This register is used to define the address and size information for boot-time access to the optional Flash memory. This register returns a zero value for functions without an Expansion ROM window and for dummy functions.

The Expansion ROM BAR is disabled through the PFPCI_CNF.*EXROM_DIS* register field.

Field	Bits	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** The number of bits that are not hard-wired to 0b is determined by the value of the GLPCI_LBARCTRL.*EXROM_SIZE* register field, loaded from the NVM.

14.3 Capabilities in PCI Configuration Space

The first entry of the PCI capabilities link list is pointed to by the Cap_Ptr register. Table 14-11 lists the capabilities supported by the E810 that reside in the PCI configuration space.

Table 14-11. PCI Capabilities List

Address Range	Item	Cases Where Capability Does Not Exist	Next Pointer	Section Reference
0x40 - 0x4F	Power Management	• None (always exists).	0x50 / 0xA0	14.3.1
0x50 - 0x6F	MSI	• Dummy function.	0x70 / 0xA0	14.3.2
0x70 - 0x8F	MSI-X	• PFPCI_CNF.MSI_EN is 0b. • Dummy function.	0xA0	14.3.3
0xA0 - 0xDF	PCIe	• None (always exists).	0xE0 / 0x00	14.3.5
0xE0 - 0xEF	VPD	• VPD Enable bit (GLPCI_CAPCTRL.VPD_EN) is cleared. • Dummy function.	0x00	14.3.4

14.3.1 PCI Power Management Capability

Table 14-12. PCI Power Management Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x40	Power Management Capabilities		Next Pointer	Capability ID (0x01)
0x44	Data	Bridge Support Extensions	Power Management Control and Status	

Table 14-13 lists the sharing of the Power Management Capability registers among the different PCI functions.

Table 14-13. Sharing the Power Management Capability Registers

Field	Sub-Field	Shared?	Replicated?	Comments	Section Reference
Capability ID		X			14.3.1.1
Next Pointer			X		14.3.1.2
Power Management Capabilities	PME_Support	X			14.3.1.3
	D2_Support	X			
	D1_Support	X			
	AUX Current	X			
	DSI	X			
	PME Clock	X		Hard-wired to 0b.	
	Version	X			

Table 14-13. Sharing the Power Management Capability Registers [continued]

Field	Sub-Field	Shared?	Replicated?	Comments	Section Reference
Power Management Control / Status	PME_Status		X		14.3.1.4
	Data_Scale	X			
	Data_Select		X		
	PME_En		X		
	No_Soft_Reset	X			
	PowerState		X		
Data Register			X		14.3.1.6

14.3.1.1 Capability ID Register (0x40; RO)

This field equals 0x01, indicating the linked list item as being the PCI Power Management registers.

14.3.1.2 Next Pointer Register (0x41; RO)

This field provides an offset to the next capability item in the capability list. See Table 14-11 for possible values of the next pointer register.

14.3.1.3 Power Management Capabilities - PMCR (0x42; RO)

This register describes the device functionality during the power management states as listed in the following table.

Bits	Init.	Type	Description
2:0	011b	RO	**Version** The E810 complies with the PCI PM specification revision 1.2.
3	0b	RO	**PME_Clock** Disabled. Hard-wired to 0b.
4	0b	RO	**Immediate Readiness** Immediate_Readiness_on_Return_to_D0 (not supported in the E810).
5	1b	RO	**DSI** The E810 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 E810 does not support the D1 state.
10	0b	RO	**D2_Support** The E810 does not support the D2 state.
15:11	0x0	RO	**PME_Support** This 5-bit field indicates the power states in which the function can generate a PME event. If GLPCI_CAPSUP.*WAKEUP_EN* is set, the condition functionality values are as follows: 01001b = No AUX Pwr PME at D0 and D3hot 11001b = AUX Pwr PME at D0, D3hot, and D3cold Otherwise zero. ***Note:*** For dummy function, this field is RO - zero.

14.3.1.4 Power Management Control/Status Register - PMCSR (0x44; RW)

This register is used to control and monitor power management events in the device.

Bits	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 E810 does not perform an internal reset upon transition from D3hot to D0 via software control of the *PowerState* bits. Configuration context is maintained when performing the soft reset. Upon transition from the D3hot to the D0 state, an initialization sequence is not needed to return the E810 to the D0 Initialized state.
7:4	0x0	RO	Reserved.
8	0b (at power up)	RWS	**PME_En** 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 *Data_Scale* field.
14:13	00b	RO	**Data_Scale** This field indicates the scaling factor that is used when interpreting the value of the Data register. This field is loaded from the NVM (through the GLPCI_PWRDATA.*DATA_SCALE* register) for legal values of Data_Select [0, 3, 4, 7, (and 8 for function 0)]. The normal value is 01b (indicating 0.1 watt/units). Reserved (00b) for any other values of Data_Select.
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 *PME_En* bit. Writing a 1b clears this bit.

14.3.1.5 PMCSR_BSE Bridge Support Extensions Register (0x46; RO)

This register is not implemented in the E810; values set to 0x00.

14.3.1.6 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, while the power scale is reported in the *Data_Scale* field in the PMCSR. The data for this field is loaded from the NVM with a default value of 0x00. It is exposed through the GLPCI_PWRDATA register.

The values for E810 functions are as follows (the relevant column is selected based on the value of the *Data_Select* field):

Field	D0 (Consume/ Dissipate)	D3 (Consume/ Dissipate)	Common
Data_Select	(0x0/0x4)	(0x3/0x7)	(0x8)
Function 0	Loaded from NVM	Loaded from NVM	Multi-function value: Loaded from the NVM Single-function value: 0x00
Other functions	Loaded from NVM	Loaded from NVM	0x00

Note: For other *Data_Select* values the Data register output is reserved (0x00).

14.3.2 MSI Capability

This structure is enabled when the PFPCI_CNF.*MSI_EN* bit is set for the function.

Table 14-14. MSI Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x50	Message Control (0x0080)		Next Pointer	Capability ID (0x05)
0x54	Message Address			
0x58	Message Upper Address			
0x5C	Reserved		Message Data	
0x60	Mask Bits			
0x64	Pending Bits			

Table 14-15 lists configuration sharing of the MSI Capability registers among the different PCI functions.

Table 14-15. Configuration Sharing of the MSI Capability

Field	Sub-Field	Shared?	Replicated?	Comments	Section Reference
Capability ID		X			14.3.2.1
Next Pointer			X		14.3.2.2
Message Control	MSI Enable		X		14.3.2.3
	Multiple Messages Capable	X			
	Multiple Message Enable		X		
	64-bit Capable	X			
	MSI per-vector masking	X			
Message Address Low			X		14.3.2.4
Message Address High			X		14.3.2.5
Message Data			X		14.3.2.6
Mask Bits			X		14.3.2.7
Pending Bits			X		14.3.2.8

14.3.2.1 Capability ID Register (0x50; RO)

This field equals 0x05, indicating that the linked list item as being the MSI registers.

14.3.2.2 Next Pointer Register (0x51; RO)

This field provides an offset to the next capability item in the capability list. See Table 14-11 for possible values of the next pointer register.

14.3.2.3 Message Control Register (0x52; RW)

Bits	Init.	Type	Description
0	0b	RW	**MSI Enable** 1b = Message Signaled Interrupts — The E810 generates an MSI for interrupt assertion instead of INTx signaling.
3:1	000b	RO	**Multiple Messages Capable** The E810 indicates a single requested message per function.
6:4	000b	RW	**Multiple Message Enable** Software writes to this field to indicate the number of allocated vectors. Since the E810 requests a single vector in the *Multiple Message Capable* field, software is expected to write 000b to this field.
7	1b	RO	**64-bit Capable** 1b = Indicates that the E810 is capable of generating 64-bit message addresses.
8	1b[1]	RO	**MSI per-vector masking** 0b = Indicates that the E810 is not capable of per-vector masking. 1b = Indicates that the E810 is capable of per-vector masking.
15:9	0x0	RO	Reserved. Reads as 0x0.

1. Value loaded from the NVM.

14.3.2.4 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.

14.3.2.5 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.

14.3.2.6 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.

14.3.2.7 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. Because the E810 only supports one message, only bit 0 of these registers are implemented.

Bits	Init.	Type	Description
0	0b	RW	**MSI Vector 0 Mask** If set, the E810 is prohibited from sending MSI messages.
31:1	0x0	RO	Reserved.

14.3.2.8 Pending Bits Register (0x64; RW)

Bits	Init.	Type	Description
0	0b	RO	**MSI Message** If set, the E810 has a pending MSI message.
31:1	0x0	RO	Reserved.

14.3.3 MSI-X Capability

More than one MSI-X capability structure per function is prohibited, though 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.

A BAR is allocated for the MSI-X structures, described in Section 14.2.6. 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 bits.

The number of MSI-X vectors per PF (denoted as N) varies with the number of physical and virtual functions as set from NVM.

The MSI-X BAR is 32 KB long.

The location and size of the MSI-X vector table and the MSI-X pending bits table are determined as follows:

- MSI-X vector table:
  - The MSI-X table structure (Section 14.3.3.2) 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.
  - Starts at offset 0x0000 from start of BAR.
  - Contains the MSI-X vectors for the PF. The number of entries in the table (N) is set from NVM. The maximum value of N is 2048 per PF.
  - The vectors start with the Vector 0 (one per PF), followed by the other vectors allocated to the PF.

- MSI-X Pending Bits table:
  - The PBA structure (Section 14.3.3.2.2) 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.
  - Starts at offset 0x8000 (32 KB) from start of BAR.
  - Contains the pending bits for the PF. The PF can be allocated a multiple of 64-bit registers. The total number of 32-bit registers is per the number of MSI-X vectors. The maximum number of registers is 8 (for the case of 256 vectors).
  - The bits start with the Vector 0 bit (one per PF), followed by bits for the other vectors allocated to the PF.

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.

14.3.3.1 Capability Structure

Table 14-16. MSI-X Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x70	Message Control (0x00090)		Next Pointer	Capability ID (0x11)
0x74	Table Offset			
0x78	PBA Offset			

Table 14-17 lists configuration sharing of the MSI-X capability registers among the different PCI functions.

Table 14-17. Configuration Sharing of the MSI-X Capability

Field	Sub-Field	Shared?	Replicated?	Comments	Section Reference
Capability ID		X			14.3.3.1.1
Next Pointer			X		14.3.3.1.2
Message Control	Table Size		X		14.3.3.1.3
Function Mask			X		
MSI-X Enable			X		
MSI-X Table Offset	Table BIR		X		14.3.3.1.4
	Table Offset		X		

Table 14-17. Configuration Sharing of the MSI-X Capability [continued]

Field	Sub-Field	Shared?	Replicated?	Comments	Section Reference
MSI-X Pending Bit Array	PBA BIR		X		14.3.3.1.5
	PBA Offset		X		
MSI-X Table			X		14.3.3.2.1
MSI-X PBA Structure			X		14.3.3.2.2

14.3.3.1.1 Capability ID Register (0x70; RO)

This field equals 0x11, indicating that the linked list item as being the MSI-X registers.

14.3.3.1.2 Next Pointer Register (0x71; RO)

This field provides an offset to the next capability item in the capability list. See Table 14-11 for possible values of the next pointer register.

14.3.3.1.3 Message Control Register (0x72; RW)

Bits	Init.	Type	Description
10:0	0xFF (256 vectors)	RO	**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. It is reflected in the GLPCI_CNF2.*MSI_X_PF_N* CSR 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 *Mask* 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 *Mask* bit states. Setting or clearing the MSI-X *Function Mask* bit has no effect on the state of the per-vector *Mask* 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 *Enable* 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.

14.3.3.1.4 MSI-X Table Offset Register (0x74; RW)

Bits	Init.	Type	Description
2:0	0x3	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 0x10:0x24, respectively.
31:3	0x000	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 *Table BIR* bits are masked off (set to 0b) by software to form a 32-bit QWord-aligned offset. ***Note:*** This field is read only.

14.3.3.1.5 MSI-X Pending Bit Array - PBA Offset (0x78; RW)

Bits	Init.	Type	Description
2:0	0x3	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 0x10:0x24, respectively.
31:3	0x200	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 *PBA BIR* bits are masked off (set to 0b) by software to form a 32-bit QWord-aligned offset. ***Note:*** This field is read only.

14.3.3.2 PF MSI-X Table Structure

The MSI-X table is made of two structures:

- MSI-X Vector Table
- MSI-X Pending Bits Table

Both are described in the sections that follow and in more detail in Chapter 13.

14.3.3.2.1 MSI-X Vector Table

DWord3 - MSIXTVCTRL	DWord2 - MSIXTMSG	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	(N-1)	Base + (N-1)*16

14.3.3.2.2 MSI-X Pending Bits Table

Field	Bits	Init.	Description
PENBIT	31:0	0x0	**MSI-X Pending Bits** Each bit is set to 1b when the appropriate interrupt request is set and cleared to 0b when the appropriate interrupt request is cleared. See Section 9.1.1.3 for more details.

14.3.4 VPD Capability

The E810 supports access to a VPD structure stored in the NVM using the following set of registers. A single VPD structure is provided, accessible through any of the physical functions.

Initial values of the configuration registers are marked in parenthesis.

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0xE0	VPD Address		Next Pointer	Capability ID (0x03)
0xE4	VPD Data			

14.3.4.1 Capability ID Register (0xE0; RO)

This field equals 0x3, indicating the linked list item as being the VPD registers.

14.3.4.2 Next Pointer Register (0xE1; RO)

Offset to the next capability item in the capability list. See Table 14-11 for possible values of the next pointer register.

14.3.4.3 VPD Address Register (0xE2; 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.

Bits	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.

14.3.4.4 VPD Data Register (0xE4; RW)

VPD read/write data.

Bits	Init.	Type	Description
31:0	X	RW	**VPD Data** VPD data can be read or written through this register. The LSB 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.

14.3.5 PCIe Capability Structure

The E810 implements the PCIe capability structure linked to the legacy PCI capability list for endpoint devices as follows:

Table 14-18. PCIe Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0xA0	PCIe Capabilities Register (0x0002)		Next Pointer	Capability ID (0x10)
0xA4	Device Capabilities			
0xA8	Device Status		Device Control	
0xAC	Link Capabilities			
0xB0	Link Status		Link Control	
0xB4	Reserved			
0xB8	Reserved		Reserved	
0xBC	Reserved			
0xC0	Reserved		Reserved	
0xC4	Device Capabilities 2			
0xC8	Reserved		Device Control 2	
0xCC	Link Capabilities 2 Register			
0xD0	Link Status 2		Link Control 2	
0xD4	Reserved			
0xD8	Reserved		Reserved	

Table 14-19 lists configuration sharing of the PCIe Capability registers among the different PCI functions.

Table 14-19. Configuration Sharing of the PCIe Capability

Field	Sub-Field	Shared?	Replicated?	Comments	Section Reference
Capability ID		X			14.3.5.1
Next Pointer			X		14.3.5.2
PCIe Capabilities		X			14.3.5.3
Device Capabilities	Max Payload Size Supported	X			14.3.5.4
	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			

Table 14-19. Configuration Sharing of the PCIe Capability [continued]

Field	Sub-Field	Shared?	Replicated?	Comments	Section Reference
Device Control	Correctable Error Reporting Enable		X		14.3.5.5
	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 mode, 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		14.3.5.6
	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			14.3.5.7
	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 14-19. Configuration Sharing of the PCIe Capability [continued]

Field	Sub-Field	Shared?	Replicated?	Comments	Section Reference
Link Control	Active State Link PM Control		X	Same policy for all PFs (Logical AND of the PF's bits). In ARI mode, use value in Function 0.	14.3.5.8
	Read Completion Boundary (RCB)		X		
	Common Clock Configuration		X	Same policy for all PFs (Logical AND of the PF's bits) In ARI mode, 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			14.3.5.9
	Negotiated Link Width	X			
	Slot Clock Configuration	X			
Device Capabilities 2	Completion Timeout Ranges Supported	X			14.3.5.10
	Completion Timeout Disable 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.	14.3.5.11
	Completion Timeout Disable		X	Completion timeout mechanism enabled per PF.	
	IDO Request Enable		X		
	IDO Completion Enable		X		
	OBFF Enable		X	PF0 only. RsvdP on other functions.	
Link Capabilities 2		X			14.3.5.12
Link Control 2		X		PF0 only. RsvdP on other functions.	14.3.5.13
Link Status 2		X			14.3.5.14

14.3.5.1 Capability ID Register (0xA0; RO)

This field equals 0x10, indicating that the linked list item as being the PCIe Capabilities registers.

14.3.5.2 Next Pointer Register (0xA1; RO)

Offset to the next capability item in the capability list. See Table 14-11 for possible values of the next pointer register.

14.3.5.3 PCIe Capabilities Register (0xA2; RO)

The PCIe Capabilities register identifies PCIe device type and associated capabilities.

Bits	Init.	Type	Description
3:0	0x2	RO	**Capability Version** Indicates the PCIe capability structure version. The E810 supports PCIe version 2 (loaded from NVM). It is reflected in the GLPCI_CAPSUP register.
7:4	0x0/0x9	RO	**Device/Port Type** Indicates the type of PCIe functions. All functions are native PCI functions with a value of 0x0 if *strap_gbe_pci_endpoint_type* strap is set to native, and Root Complex Integrated Endpoint is set to RCIE.
8	0b	RO	**Slot Implemented** The E810 does not implement slot options. Therefore, this field is hard-wired to 0b.
13:9	0x0	RO	**Interrupt Message Number** This field is hard-wired to 0x0 and is assumed to be irrelevant for endpoints.
15:14	00b	RO	Reserved.

14.3.5.4 Device Capabilities Register (0xA4; RO)

This register identifies the PCIe device-specific capabilities.

Bits	Init.	Type	Description
2:0	010b	RO	**Max Payload Size Supported** This field indicates the maximum payload that the E810 can support for TLPs. Set according to the *strap_gbe_imps* strap. If *strap_gbe_imps* strap = 111b, the default is 000b (128 bytes)
4:3	00b	RO	**Phantom Function Supported** Not supported by the E810.
5	1b	RO	**Extended Tag Field Supported** Maximum supported size of the *Tag* field. The E810 supports an 8-bit *Tag* field for all functions.
8:6	011b	RO	**Endpoint L0s Acceptable Latency** This field indicates the acceptable latency that the E810 can withstand due to the transition from L0s state to the L0 state. All functions share the same value loaded from the NVM. The default value of 011b denotes a maximum 512 ns.
11:9	110b	RO	**Endpoint L1 Acceptable Latency** This field indicates the acceptable latency that the E810 can withstand due to the transition from L1 state to the L0 state. The default value of 110b denotes a maximum of 64 µs. All functions share the same value loaded from the NVM. It is reflected in the GLPCI_PMSUP register.
12	0b	RO	Hard-wired in the E810 to 0b for all functions.
13	0b	RO	Hard-wired in the E810 to 0b for all functions.
14	0b	RO	Hard-wired in the E810 to 0b for all functions.
15	1b	RO	**Role Based Error Reporting** Hard-wired in the E810 to 1b for all functions.
17:16	00b	RO	Reserved.
25:18	0x0	RO	**Slot Power Limit Value** Hard-wired in the E810 to 0x0 for all functions.
27:26	00b	RO	**Slot Power Limit Scale** Hard-wired in the E810 to 0b for all functions.
28	1b	RO	**Function Level Reset Capability** A value of 1b indicates the Function supports the optional Function Level Reset (FLR) mechanism.

Bits	Init.	Type	Description
31:29	000b	RO	Reserved.

14.3.5.5 Device Control Register (0xA8; RW)

This register controls the PCIe-specific parameters.

Bits	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 E810 is permitted to set the *Relaxed Ordering* bit in the *Attribute* field of write transactions that do not need strong ordering. Refer to Section 3.1.2.7.2 for more details.
7:5	000b (128 bytes)	RW	**Max Payload Size** This field sets the maximum TLP payload size for E810 functions. As a receiver, the E810 must handle TLPs as large as the set value. As a transmitter, the E810 must not generate TLPs exceeding the set value. In ARI mode, *Max Payload Size* 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 E810 uses 8-bit tags when this bit is set, and a 5-bit tag when disabled.
9	0b	RO	**Phantom Functions Enable** Not implemented in the E810.
10	0b	RWS	**Auxiliary Power PM Enable** When set, enables the E810 to draw AUX power independent of PME AUX power. The E810 is a multi-function device, and is therefore allowed to draw AUX power if at least one of the functions has this bit set.
11	0b	RO	**Enable No Snoop** Hard-wired to 0b, as the E810 never sets the no snoop attribute in a TLP.
14:12	010b	RW	**Max Read Request Size** This field sets maximum read request size for the E810 as a requester. 000b = 128 bytes 001b = 256 bytes 010b = 512 bytes 011b = 1024 bytes 100b = 2048 bytes 101b = 4096 bytes 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.

14.3.5.6 Device Status Register (0xAA; RW1C)

This register provides information about PCIe device-specific parameters.

Bits	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 E810 received an unsupported request. This field is separate per PF. However, in the case where an error cannot be associated with a PF, this bit is set in all PFs and VFs.
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 E810 has ANY transactions pending. Transactions include completions for any outstanding non-posted requests for all used traffic classes.
6	0b	RO	**Emergency Power Reduction Detected** Not supported in the E810.
15:7	0x0	RsvdZ	Reserved.

14.3.5.7 Link Capabilities Register (0xAC; RO)

This register identifies PCIe link-specific capabilities.

Bits	Init.	Type	Description
3:0	0x4	RO	**Maximum Link Speed** This field indicates the supported Link speed of the associated port. The encoding is the binary value of the bit location in the supported link speeds vector (in the Link Capabilities 2 register) that corresponds to the maximum link speed. For example, a value of 0011b in this field indicates that the maximum link speed is that corresponding to bit 2 in the supported link speeds vector, which is 8.0 GT/s. Multi-function devices associated with an upstream port must report the same value in this field for all functions.
9:4	0x10	RO	**Maximum Link Width** Indicates the maximum link width. The E810 supports x1, x4, x8, and x16 link width. This field is loaded from the NVM and is reflected in the GLPCI_LINKCAP register. Defined encoding: 000000b = Reserved 000001b = x1 000010b = Reserved 000100b = x4 001000b = x8 001100b = x12 010000b = x16 100000b = x32

Bits	Init.	Type	Description
11:10	10b	RO	**Active State Link PM Support** Indicates the level of the active state of power management supported in the E810. 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.
14:12	100b	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 µs 101b = 1 µs – 2 µs 110b = 2 µs – 4 µs 111b = Reserved. All functions share the same value loaded from the NVM.
17:15	010b	RO	**L1 Exit Latency** Indicates the exit latency from L1 to L0 state. 000b = Less than 1 µs 001b = 1 µs - 2 µs 010b = 2 µs - 4 µs 011b = 4 µs - 8 µs 100b = 8 µs - 16 µs 101b = 16 µs - 32 µs 110b = 32 µs - 64 µs 111b = More than 64 µs All functions share the same value loaded from the NVM.
18	0b	RO	**Clock Power Management** Not supported.
19	0b	RO	**Surprise Down Error Reporting Capable** Hard-wired to 0b.
20	0b	RO	**Data Link Layer Link Active Reporting Capable**
21	0b	RO	**Link Bandwidth Notification Capability** Hard-wired to 0b (not applicable to end-points).
22	1b	HwInit	**ASPM Optionality Compliance** Software is permitted to use the value of this bit to help determine whether to enable ASPM or to 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. The field is set through the *gbe_DeviceNumberN* strap.

14.3.5.8 Link Control Register (0xB0; RO)

This register controls PCIe link-specific parameters.

Bits	Init.	Type	Description
1:0	00b	RW	**Active State Link PM Control** This field controls the active state PM enabled on the link. Link PM functionality is determined by the lowest common denominator of all functions. Defined encodings are: 00b = PM Disabled. 01b = L0s Entry Supported. 10b = L1 Entry Enabled. 11b = L0s and L1 Supported. In ARI mode, 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	RsvdP	Reserved.
3	0b	RW	**Read Completion Boundary**
4	0b	RO	**Link Disable** Reserved for endpoint devices. Hard-wired to 0b.
5	0b	RO	**Retrain Link** Not applicable for endpoint devices. Hard-wired to 0b.
6	0b	RW	**Common Clock Configuration** When set, indicates that the E810 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. This parameter affects the L0s exit latencies. In ARI mode, 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 the transmission of additional ordered sets when exiting the L0s state and when in the recovery state. For multi-function devices, if any function has this bit set, the component must transmit the additional ordered sets when exiting L0s or when in recovery.
8	0b	RO	**Enable Clock Power Management** Not supported. Hard-wired to 0b.
9	0b	RO/RsvdP	**Hardware Autonomous Width Disable** When set to 1b, this bit disables hardware from changing the link width for reasons other than attempting to correct an unreliable link operation by reducing link width. Not supported - Function 0 is hard-wired to 0b (RO). Other functions are RsvdP.
10	0b	RO	**Link Bandwidth Management Interrupt Enable** Not applicable to endpoints. Hard-wired to 0b.
11	0b	RO	**Link Autonomous Bandwidth Interrupt Enable** Not applicable to endpoints. Hard-wired to 0b.
13:12	00b	RsvdP	Reserved.
15:14	00b	RO	**DRS Signaling Control** Not applicable to endpoints. Hard-wired to 0b.

14.3.5.9 Link Status Register (0xB2; RO)

This register provides information about PCIe link-specific parameters.

Bits	Init.	Type	Description
3:0	Undefined	RO	**Current Link Speed** This field indicates the negotiated link speed of the given PCIe link. The encoded value specifies a bit location in the supported link speeds vector (in the Link Capabilities 2 register) that corresponds to the current link speed. For example, a value of 0010b in this field indicates that the current Link speed is that corresponding to bit 1 in the supported link speeds vector, which is 5.0 GT/s.
9:4	Undefined	RO	**Negotiated Link Width** Indicates the negotiated width of the link. Relevant encodings for the E810 are: 000001b = x1 000010b = x2 000100b = x4 001000b = x8 010000b = x16
10	0b	RO	Undefined.
11	0b	RO	**Link Training** Indicates that link training is in progress. This field is not applicable, is reserved for endpoint devices, and is hard-wired to 0b.
12	1b	HwInit	**Slot Clock Configuration** When set, indicates that the E810 uses the physical reference clock that the platform provides at the connector. This bit must be cleared if the E810 uses an independent clock. The *Slot Clock Configuration* bit is loaded from the NVM.
13	0b	RO	**Data Link Layer Link Active** Not supported in the E810. Hard-wired to 0b.
14	0b	RO	**Link Bandwidth Management Status** Not supported in the E810. Hard-wired 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.

14.3.5.10 Device Capabilities 2 Register (0xC4; RO)

This register identifies the PCIe device-specific capabilities.

Bits	Init.	Type	Description
3:0	0x3	RO	**Completion Timeout Ranges Supported** This field indicates the E810'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 E810 supports. 0000b = Completion timeout programming not supported. The E810 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** ***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	0b	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 NVM as 0b (LTR is not supported by this product). It is reflected in the GLPCI_CAPSUP register.
13:12	00b	RO	**TPH Completer Supported** Value indicates Completer support for TPH or Extended TPH. This capability is not supported.
15:14	00b	R0	**LN System CLS** Applicable only to Root Ports and RCRBs. Must be 00b for all other Function types.
16	1b	RO	**10-Bit Tag Completer Supported** 0b = The Function does not support 10-Bit Tag Completer capability. 1b = The Function supports 10-Bit Tag Completer capability. Supported by the E810.
17	0b	RO	**10-Bit Tag Requester Supported** 0b = The Function does not support 10-Bit Tag Requester capability. 1b = The Function supports 10-Bit Tag Requester capability. Not supported by the E810.

Bits	Init.	Type	Description
19:18	00b	HwInit	**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. Loaded from NVM with a value of 00b (OBBF is not supported in this product). The value loaded from NVM is reflected in the GLPCI_PMSUP register.
20	1b	RO	**Extended Fmt Field Supported** 0b = The Function supports a 2-bit definition of the *Fmt* field. 1b = The Function supports the 3-bit definition of the *Fmt* field.
21	1b	RO	**End-End TLP Prefix Supported** Indicates whether End-End TLP Prefix support is offered by a Function.
23:22	1b	RsvdP	**Max End-End TLP Prefixes** Indicates the maximum number of End-End TLP Prefixes supported by this Function.
25:24	00b	RO	**Emergency Power Reduction Supported** Not supported in the E810.
26	0b	RO	**Emergency Power Reduction Initialization Required** N/A for the E810.
31:27	0x0	RsvdP	Reserved.

14.3.5.11 Device Control 2 Register (0xC8; RW)

This register controls the PCIe-specific parameters.

Bits	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: 50 µs to 50 ms. ***Note:*** It is strongly recommended that the completion timeout mechanism not expire in less than 10 ms. Values available if Range A (50 µs to 10 ms) programmability range is supported: 0001b = 50 µs to 100 µs. 0010b = 1 ms to 10 ms. Values available if Range B (10 ms to 250 ms) programmability range is supported: 0101b = 16 ms to 55 ms. 0110b = 65 ms to 210 ms. Values available if Range C (250 ms to 4 s) programmability range is supported: 1001b = 260 ms to 900 ms. 1010b = 1 s to 3.5 s. Values available if the Range D (4 s to 64 s) programmability range is supported: 1101b = 4 s to 13 s. 1110b = 17 s to 64 s. Values not defined 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. Specifically, FLR clears this field to its default, so that completions are expected to return by the default time. ***Note:*** For dummy function, this field is RO - zero.

Bits	Init.	Type	Description
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.
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.
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
10	0b	RO	**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 LTR is not supported, this bit is RO with a value of 0b.
11	00b	RO	**Emergency Power Reduction Request** Not supported by the E810.
12	0b	RW	**10-bit Tag Requester Enable**
14:13	00b	RW/RsvdP	**OBFF Enable** 00b = Disabled. 01b = Enabled using Message signaling [Variation A]. 10b = Enabled using Message signaling [Variation B]. 11b = Enabled using WAKE# signaling. For a multi-function device, the field in Function 0 is of type RW, and only Function 0 controls the component's behavior. In all other Functions of that device, this field is of type RsvdP.
15	0b	RsvdP	**End-End TLP Prefix Blocking** Not applicable to endpoints (RsvdP).

14.3.5.12 Link Capabilities 2 Register (0xCC; RO)

Bits	Init.	Type	Description
0	0b	RsvdP	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 GLPCI_LINKCAP register.
8	0b	RO	**Crosslink Supported** When set to 1b, this bit indicates that the associated Port supports crosslinks. It is recommended that this bit be Set in any Port that supports crosslinks even though doing so is only required for Ports that also support operating at 8.0 GT/s or higher Link speeds.

Bits	Init.	Type	Description
0	0b	RsvdP	Reserved.
15:9	0x0	RO	**Lower SKP OS Generation Supported Speeds Vector** If this field is non-zero, it indicates that the Port, when operating at the indicated speed(s), supports SRIS and also supports software control of the SKP Ordered Set transmission scheduling rate. Bit definitions within this field are: Bit 0 = 2.5 GT/s Bit 1 = 5.0 GT/s Bit 2 = 8.0 GT/s Bit 3 = 16.0 GT/s Bits 6:4 = RsvdP
22:6	0x0	RO	**Lower SKP OS Reception Supported Speeds Vector** If this field is non-zero, it indicates that the Port, when operating at the indicated speed(s), supports SRIS and also supports receiving SKP OS at the rate defined for SRNS while running in SRIS. Bit definitions within this field are: Bit 0 = 2.5 GT/s Bit 1 = 5.0 GT/s Bit 2 = 8.0 GT/s Bit 3 = 16.0 GT/s Bits 6:4 = RsvdP
23	1b	RO	**Re-timer Presence Detect Supported** When set to 1b, this bit indicates that the associated Port supports detection and reporting of re-timer presence. This bit must be set to 1b in a Port when the *Supported Link Speeds Vector* of the Link Capabilities 2 register indicates support for a Link speed of 16.0 GT/s or higher. It is permitted to be set to 1b regardless of the supported Link speeds.
24	1b	RO	**Two Re-timers Presence Detect Supported** When set to 1b, this bit indicates that the associated Port supports detection and reporting of two re-timers presence. This bit must be set to 1b in a Port when the *Supported Link Speeds Vector* of the Link Capabilities 2 register indicates support for a Link speed of 16.0 GT/s or higher. It is permitted to be set to 1b regardless of the supported Link speeds if the *Re-timer Presence Detect Supported* bit is also set to 1b.
30:25	0x0	RsvdP	Reserved.
31	0b	RO	**DRS Supported** When set, indicates support for the optional Device Readiness Status (DRS) capability. Not supported by the E810.

14.3.5.13 Link Control 2 Register (0xD0; RWS)

Bits	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 *Enter Compliance* 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 1 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 *Supported Link Speeds* field, the result is undefined. The default value of this field is the highest link speed supported by the E810 (as reported in the *Supported Link Speeds* field of the Link Capabilities register).

Bits	Init.	Type	Description
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 *Target Link Speed* 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.
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 *Enter Compliance* 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 *Enter Compliance* 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.

14.3.5.14 Link Status 2 Register (0xD2; RW)

Bits	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 not operating at 5.0 GT/s speed. Encodings: 0b = -6 dB 1b = -3.5 dB ***Note:*** Same value must be reported for all functions.
1	0b	ROS/RsvdZ	**Equalization Complete** When set to 1b, this bit indicates that the Transmitter Equalization procedure has completed. ***Note:*** This bit must be implemented in Function 0 and RsvdZ in other Functions.
2	0b	ROS/RsvdZ	**Equalization Phase 1 Successful** When set to 1b, this bit indicates that Phase 1 of the Transmitter Equalization procedure has successfully completed. ***Note:*** This bit must be implemented in Function 0 and RsvdZ in other Functions.
3	0b	ROS/RsvdZ	**Equalization Phase 2 Successful** When set to 1b, this bit indicates that Phase 2 of the Transmitter Equalization procedure has successfully completed. ***Note:*** This bit must be implemented in Function 0 and RsvdZ in other Functions.
4	0b	ROS/RsvdZ	**Equalization Phase 3 Successful** When set to 1b, this bit indicates that Phase 3 of the Transmitter Equalization procedure has successfully completed. ***Note:*** This bit must be implemented in Function 0 and RsvdZ in other Functions.
5	0b	RW1C/RsvdZ	**Link Equalization Request** This bit is Set by hardware to request the Link equalization process to be performed on the Link. ***Note:*** This bit must be implemented in Function 0 and RsvdZ in other Functions.
6	0b	RWS (Func 0) RsvdZ (else)	**Re-timer Presence Detected**
7	0b	RWS (Func 0) RsvdZ (else)	**Two Re-timer Presence Detected**
9:8	00b	RO	**Crosslink Resolution**
15:10	0x0	RsvdZ	Reserved.

14.4 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 E810 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 E810 supports the following PCIe extended capabilities:

Table 14-20. Extended Capabilities List

Address Range	Item	Cases Where Capability Does Not Exist	Next Pointer	Section Reference
0x100 - 0x144	Advanced Error Reporting (AER)	• None (always present).	Any of the below / 0x000	14.4.1
0x150 - 0x158	Device Serial Number	• NVM is not valid.	Any of the below / 0x000	14.4.2
0x148 - 0x14C	Alternative RID Interpretation (ARI)	• ARI Enabled bit in NVM is set to 0b	Any of the below / 0x000	14.4.3
0x160 - 0x19C	Single Root I/O Virtualization (SR-IOV)	• The global SR-IOV Enable bit in NVM is set to 0b (exposed via the GLPCI_CAPSUP.*IOV_EN* bit). • This is a dummy function. • The per-PF SR-IOV Enable bit is set to 0b (PF_VT_PFALLOC.*VALID* is cleared).	Any of the below / 0x000	14.4.4
0x1A0 - 0x1A8	TPH Requester	• TPH Enabled bit in NVM is set to 0b. • This is a dummy function.	Any of the below / 0x000	14.4.5
0x1B0 - 0x1B4	Access Control Services (ACS)	• ACS Enabled bit in NVM is set to 0b. • A single PF is enabled and SR-IOV is disabled.	Any of the below / 0x000	14.4.6
0x1C0 - 0x1C4	Reserved.			
0x1D0 - 0x1FA	Secondary PCI Express	• Secondary PCIe Enabled bit in NVM is set to 0b. • Function is not Function 0.	0x200 / 0x000	14.4.7
0x200 - 0x208	Data Link Feature	• GLPCI_CAPSUP.*DLFE_EN* is cleared.	0x210 / 0x000	14.4.8
0x210 - 0x23C	Physical Layer 16.0 GT/s Extended Capability	• GLPCI_CAPSUP.GEN4_EXT_EN is cleared. • Function is not function 0.	0x250 / 0x000	14.4.10
0x250 - 0x294	Lane Margining at the Receiver Capability	• GLPCI_CAPSUP.GEN4_MARG_EN is cleared. • Function is not function 0.	0x2D0 / 0x000	14.4.11
0x2D0 - 0x2D4	PASID	• GLPCI_CAPSUP.PASID_EN is cleared. • This is a dummy function.	0x000	14.4.9

14.4.1 Advanced Error Reporting (AER) Capability

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 14-21. AER Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x100	Next Capability Pointer	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			
0x12C	RsvdP (Root Error Command)			
0x130	RsvdZ (Root Error Status)			
0x134	RsvdP (Error Source Identification Register)			
0x138 - 0x144	TLP Prefig Log Registers = 0x0 (not implemented)			

Table 14-22 summarizes configuration sharing of the AER Capability registers among the different PCI functions.

Table 14-22. Configuration Sharing of the AER Capability

Field	Sub-Field	Shared?	Replicated?	Comments	Section Reference
Enhanced Capability Header Register	Extended Capability ID	X			14.4.1.1
	Capability Version	X			
	Next Capability Offset		X		
Uncorrectable Error Status			X		14.4.1.2
Uncorrectable Error Mask			X		14.4.1.3
Uncorrectable Error Severity			X		14.4.1.4
Correctable Error Status			X		14.4.1.5
Correctable Error Mask			X		14.4.1.6
Advanced Error Capabilities and Control	First Error Pointer		X		14.4.1.7
	ECRC Generation Capable	X			
	ECRC Generation Enable		X	ECRC insertion is per PF.	
	ECRC Check Capable	X			
	ECRC Check Enable		X	See Section 3.1.2.9.	
Header Log			X		14.4.1.8

14.4.1.1 Advanced Error Reporting Enhanced Capability Header Register (0x100; RO)

Bits	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 14-20 for possible values of the next capability offset.

14.4.1.2 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.

Bits	Init.	Type	Description
0	0b	RO	Reserved.
3:1	000b	RsvdZ	Reserved.
4	0b	RW1CS	**Data Link Protocol Error Status**
5	0b	RO	**Surprise Down Error Status**
11:6	0x0	RsvdZ	Reserved.
12	0b	RW1CS	**Poisoned TLP Received 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.
22	0b	RO	**Uncorrectable Internal Error Status** Not supported. Hard-wired to 0b.
23	0b	RO	**Multicast Blocked TLP Error Status** Not supported. Hard-wired to 0b.
24	0b	RO	**AtomicOp Egress Blocked Status** Not supported. Hard-wired to 0b.
25	0b	RO	**TLP Prefix Blocked Error Status** Not supported. Hard-wired to 0b.
26	0b	RO	**Poisoned TLP Egress Blocked Error Status** Not supported. Hard-wired to 0b.
31:27	0x0	RsvdP	Reserved.

14.4.1.3 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.

Bits	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 Received 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** Not supported. Hard-wired to 0b.
22	0b	RO	**Uncorrectable Internal Error Mask** Not supported. Hard-wired to 0b.
23	0b	RO	**Multicast Blocked TLP Error Mask** Not supported. Hard-wired to 0b.
24	0b	RO	**AtomicOp Egress Blocked Mask** Not supported. Hard-wired to 0b.
25	0b	RO	**TLP Prefix Blocked Error Mask** Not supported. Hard-wired to 0b.
26	0b	RO	**Poisoned TLP Egress Blocked Error Mask** Not supported. Hard-wired to 0b.
31:27	0x0	RsvdP	Reserved.

14.4.1.4 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.

Bits	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 Received 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**
21	0b	RO	**ACS Violation Severity** Not supported. Hard-wired to 0b.
22	0b	RO	**Uncorrectable Internal Error Severity** Not supported. Hard-wired to 0b.
23	0b	RO	**Multicast Blocked TLP Error Severity** Not supported. Hard-wired to 0b.
24	0b	RO	**AtomicOp Egress Blocked Severity** Not supported. Hard-wired to 0b.
25	0b	RO	**TLP Prefix Blocked Error Severity** Not supported. Hard-wired to 0b.
26	0b	RO	**Poisoned TLP Egress Blocked Error Severity** Not supported. Hard-wired to 0b.
31:27	0x0	RsvdP	Reserved.

14.4.1.5 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.

Bits	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**
15:14	00b	RO	Reserved.
31:16	0x0	RsvdZ	Reserved.

14.4.1.6 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.

Bits	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**
15:14	00b	RO	Reserved.

14.4.1.7 Advanced Error Capabilities and Control Register (0x118; RO)

Bits	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 GLPCI_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 GLPCI_CAPSUP register.
8	0b	RWS	**ECRC Check Enable** When 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 0.
11	0b	RsvdP	**TLP Prefix Log Present** Not supported. Hard-wired to 0b.
12	0b	RO	**Completion Timeout Prefix/Header Log Capable** Not supported in the E810.
15:13	000b	RsvdP	Reserved.

14.4.1.8 Header Log Register (0x11C - 0x128; ROS)

The Header Log register captures the header for the transaction that generated an error. This register is 16 bytes.

Bits	Init.	Type	Description
127:0	0x0	ROS	**Header Log Register** Header of the packet in error.

14.4.2 Serial Number

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.

Serial Number capability is implemented for Function 0; all other functions return the same device serial number value as that reported by Function 0.

The capability is disabled when the MAC Address in the GLPCI_SERL and GLPCI_SERH registers is 0x00...0 (indicating that the NVM is not valid or a proper MAC Address was not loaded).

Table 14-23. Serial Number Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x150	Next Capability Pointer	Version (0x1)	Serial ID Capability ID (0x0003)	
0x154	Serial Number Register (Lower DWord)			
0x158	Serial Number Register (Upper DWord)			

Table 14-24 summarizes configuration sharing of the Serial Number Capability registers among the different PCI functions.

Table 14-24. Configuration Sharing of the Serial Number Capability

Field	Sub-Field	Shared?	Replicated?	Comments	Section Reference
Enhanced Capability Header Register	Extended Capability ID	X			14.4.2.1
	Capability Version	X			
	Next Capability Offset		X		
Serial Number Registers		X		See comment above.	14.4.2.2

14.4.2.1 Device Serial Number Enhanced Capability Header Register (0x150; RO)

Bits	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 14-20 for possible values of the next capability offset.

14.4.2.2 Serial Number Registers (0x154 - 0x158; 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 0x154 holds the lower 32 bits and the register at offset 0x158 holds the higher 32 bits. The following figure details the allocation of register fields in the Serial Number register. The table that follows provides the respective bit definitions.

Bits	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 Most Significant Bit						Least Significant Byte 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 Most Significant Bit						Least Significant Byte 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 (MAC Address of 00-A0-C9-23-45-67). 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 Most Significant Bit						Least Significant Byte Least Significant Bit	

The Ethernet MAC Address for the Serial Number capability is loaded from NVM (not the same field that is loaded from NVM into the Station MAC Address registers). It is reflected in the GLPCI_SERL and GLPCI_SERH registers. In the above example:

- GLPCI_SERL = C9-23-45-67
- GLPCI_SERH = 00-00-00-A0

Note: The official document that defines EUI-64 is:
http://standards.ieee.org/regauth/oui/tutorials/EUI64.html

14.4.3 Alternate Routing ID Interpretation (ARI) Capability Structure

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 GLPCI_CAPSUP.*ARI_EN* bit is set from NVM.

This capability should not be exposed in Root Complex Integrated endpoints, and the GLPCI_CAPSUP.*ARI_EN* bit should be set accordingly.

The ARI capability structure is as follows:

Table 14-25. ARI Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x148	Next Capability Pointer	Version (0x1)	ARI Capability ID (0x000E)	
0x14C	ARI Control Register		ARI Capability Register	

Table 14-26 summarizes configuration sharing of the ARI Capability registers among the different PCI functions.

Table 14-26. Configuration Sharing of the ARI Capability

Field	Sub-Field	Shared?	Replicated?	Comments	Section Reference
Enhanced Capability Header Register	Extended Capability ID	X			14.4.3.1
	Capability Version	X			
	Next Capability Offset		X		
ARI capability Register	Next Function Pointer		X		14.4.3.2

14.4.3.1 PCIe ARI Header Register (0x148; RO)

Field	Bits	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 14-20 for possible values of the next capability offset.

14.4.3.2 PCIe ARI Capability Register (0x14C; RO)

Field	Bits	Init.	Type	Description
M	0	0b	RO	**MFVC Function Groups Capability (M)** Applicable only for Function 0. Must be 0b for all other Functions. If 1b, indicates that the ARI Device supports Function Group level arbitration via its Multi-Function Virtual Channel (MFVC) Capability structure. Not supported in the E810.
A	1	0b	RO	**ACS Function Groups Capability (A)** Applicable only for Function 0. Must be 0b for all other functions. If 1b, indicates that the ARI device supports function group level granularity for ACS P2P Egress Control via its ACS capability structures. Not supported in the E810.
Reserved	7:2	0x0	RsvdP	Reserved.

Field	Bits	Init.	Type	Description
NFN	15:8	See description[1]	RO	**Next Function Number** 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. Functions can be disabled during the Power-On-Reset flow (through strapping pins, SMASH/CLP commands, NC-SI commands) affecting this field.
M_EN	16	0b	RO	**MFVC Function Groups Enable (M)** Applicable only for Function 0. Must be hard-wired to 0b for all other Functions. When set, the ARI Device must interpret entries in its Function Arbitration Table as Function Group Numbers rather than Function Numbers. Not supported in the E810.
A_EN	17	0b	RO	**ACS Function Groups Enable (A)** Applicable only for Function 0. Must be hard-wired to 0b for all other Functions. When set, each Function in the ARI Device must associate bits within its Egress Control Vector with Function Group Numbers rather than Function Numbers. Not supported in the E810.
Reserved	19:18	00b	RO	Reserved.
FGN	22:20	000b	RO	**Function Group Number** Not supported in the E810.
Reserved	31:23	0x0	RsvdP	Reserved.

1. If function zero is a dummy function, this register should keep its attributes according to the function number. Disabled functions are skipped.

14.4.4 SR-IOV Capability Structure

This is a structure used to support the SR-IOV capabilities reporting and control. The capability is exposed when the GLPCI_CAPSUP.*IOV_EN* bit is set from NVM and the PF_VT_PFALLOC.*VALID* is set.

The following tables shows the implementation of this structure in the E810.

Table 14-27. SR-IOV Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x160	Next Capability Offset	Version (0x1)	SR-IOV Capability ID (0x0010)	
0x164	SR-IOV Capabilities			
0x168	SR-IOV Status		SR-IOV Control	
0x16C	Total VFs (RO)		Initial VFs (RO)	
0x170	Reserved	Function Dependency Link (RO)	Num VFs (RW)	
0x174	VF Stride (RO)		First VF Offset (RO)	
0x178	VF Device ID		Reserved	
0x17C	Supported Page Size (0x553)			
0x180	System Page Size (RW)			
0x184	VF BAR0 - Low (RW)			
0x188	VF BAR0 - High (RW)			
0x18C	VF BAR2 (RO)			
0x190	VF BAR3 - Low (RW)			
0x194	VF BAR3 - High (RW)			

Table 14-27. SR-IOV Capability Structure [continued]

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x198	VF BAR5 (RO)			
0x19C	VF Migration State Array Offset (RO)			

Table 14-28 summarizes configuration sharing of the SR-IOV Capability registers among the different PCI functions.

Table 14-28. Configuration Sharing of the SR-IOV Capability

Field	Sub-Field	Shared?	Replicated?	Comments	Section Reference
Enhanced Capability Header Register	Extended Capability ID	X			14.4.4.1
	Capability Version	X			
	Next Capability Offset		X		
SR-IOV Capabilities	VF Migration Capable	X		Not supported.	14.4.4.2
	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		14.4.4.3
	Memory Space Enable		X		
	ARI Capable Hierarchy	X		PF0 only. RO zero in all other functions.	
Initial VFs			X		14.4.4.4
Total VFs			X		
Num VFs			X		14.4.4.5
Function Dependency Link			X	Each PF indicates its PF number here.	
First VF Offset			X		14.4.4.6
VF Stride			X		
VF Device ID			X		14.4.4.7
Supported Page Size		X			14.4.4.8
System Page Size			X		14.4.4.9
VF BARs			X		14.4.4.10 through 14.4.4.15

14.4.4.1 PCIe SR-IOV Header Register (0x160; RO)

Field	Bits	Init.	Type	Description
ID	15:0	0x10	RO	**PCIe Extended Capability ID** PCIe extended capability ID for the SR-IOV capability.

Field	Bits	Init.	Type	Description
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 Pointer	31:20	0x0	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 14-20 for possible values of the next capability offset.

14.4.4.2 PCIe SR-IOV Capabilities Register (0x164; RO)

Field	Bits	Init.	Type	Description
VFMC	0	0b	RO	**VF Migration Capable** Migration Capable Device running under Migration Capable MR-PCIM. RO as zero in the E810. Not supported in the E810.
ARI CHP	1	1b (lowest SR-IOV-enabled function) 0b (else)	RO	**ARI Capable Hierarchy Preserved** If set, the *ARI Capable Hierarchy* bit is preserved across certain power state transitions. Only present in lowest SR-IOV-enabled function. Read-only zero in other PFs.
VF10B	2	0x0	RO	**VF 10-Bit** VF 10-Bit Tag Requester Supported - not supported.
Reserved	20:3	0x0	RO	Reserved.
IMN	31:21	0x0	RO	**VF Migration Interrupt Message Number** Indicates the MSI/MSI-X vector used for the interrupts. Not supported in the E810.

14.4.4.3 PCIe SR-IOV Control Register (0x168; RW)

Field	Bits	Init.	Type	Description
VFE	0	0b	RW	**VF Enable** Manages the assignment of VFs to the associated PF. If *VF Enable* 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 *VF Enable* 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 *VF Enable* has been cleared. Any errors already logged via PF error reporting registers, remain logged. However, no new VF errors are logged after VF Enable is cleared.
VF ME	1	0b	RO	**VF Migration Enable** Enables/Disables VF Migration Support. Not supported in the E810.
VF MIE	2	0b	RO	**VF Migration Interrupt Enable** Enables/Disables VF Migration State Change Interrupt Not supported in the E810.

Field	Bits	Init.	Type	Description
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 *Memory Space Enable* bit in a functions PCI command register. The default value for this bit is 0b. When *VF Enable* is 1b, 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 1b and *VF MSE* is zero. ***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 can choose to set VF MSE after programming the VF BARn registers, prior to setting *VF Enable* to 1b.
VF ARI	4	0b	RW (lowest SR-IOV-enabled function) ROS (else)	**ARI Capable Hierarchy** The E810 can locate VFs in function numbers 8 to 255 of the captured bus number. If either *ARI Capable Hierarchy Preserved* or *No_Soft_Reset* are set, a power state transition of this PF from D3hot to D0 does not affect the value of this bit
Reserved	15:5	0x0	RO	Reserved.
VFMIS	16	0b	RO	**VF Migration Status** Indicates a VF Migration In or Migration Out Request has been issued by MR-PCIM. To determine the cause of the event, software can scan the VF State Array. Not implemented in the E810.
Reserved	31:17	0x0	RO	Reserved.

14.4.4.4 PCIe SR-IOV Initial/Total VFs Register (0x16C; RO)

Field	Bits	Init.	Type	Description
InitialVFs	15:0	See Section 14.5.1.1	RO	**Initial VFs** Indicates the number of VFs that are initially associated with the PF. If *VF Migration Capable* is cleared, this field must contain the same value as *TotalVFs*. In the E810 this parameter is equal to the TotalVFs in this register.
TotalVFs	31:16	See Section 14.5.1.1	RO	**Total VFs** Defines the maximum number of VFs that can be associated with the PF. This field is derived from the PF_VT_PFALLOC.*FIRSTVF* and PF_VT_PFALLOC register fields loaded from NVM.

14.4.4.5 PCIe SR-IOV Num VFs Register (0x170; RW)

Field	Bits	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 and *VF Enable* is set to 1b.
FDL	23:16	0x0 (Func 0)[1] 0x1 (Func 1) ... 0xn (Func n)	RO	**Function Dependency Link** Defines dependencies between physical functions allocation. In the E810 there are no constraints.
Reserved	31:24	0x0	RO	Reserved.

1. Applies to dummy function as well.

14.4.4.6 PCIe SR-IOV VF RID Mapping Register (0x174; RO)

Field	Bits	Init.	Type	Description
FVO	15:0	0x100 + FirstVF - PF#	RO	**First VF offset** Defines the Requester 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 *VF Enable* is set. If *VF Enable* is 0b, the contents are undefined. If the *VF ARI* bit is set, this field changes to 0x8 + *FirstVF* - PF#. This field is derived from the PF_VT_PFALLOC.*FIRSTVF* register field loaded from NVM.
VFS	31:16	0x1[1]	RO	**VF stride** Defines the Requester 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 *VF Enable* is set and *NumVFs* is non-zero. If *VF Enable* is 0b or if *NumVFs* is zero, the contents are undefined.

1. See Section 14.5.1.1.

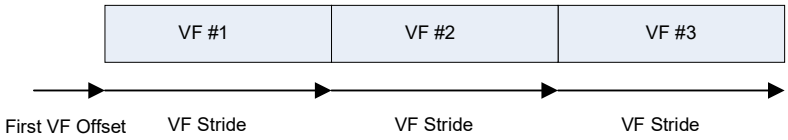


Figure 14-1. VF Stride

14.4.4.7 PCIe SR-IOV VF Device ID Register (0x178; RO)

All Virtual functions have the same default value of 0x1889, and can be auto-loaded from the NVM.

The VF Device ID is loaded from NVM according to the following rules:

- Device ID is loaded from NVM if the GLPCI_CAPSUP.*LOAD_DEV_ID* bit is set.
- The Device ID value of all VFs associated with a given PF is loaded to the respective PFPCI_DEVID.*VF_DEV_ID* field.

14.4.4.8 PCIe SR-IOV Supported Page Size Register (0x17C; RO)

Field	Bits	Init.	Type	Description
Supported Page Size	31:0	0x553	RO	**Supported Page Size** For PFs that support the stride-based BAR mechanism, this field defines the supported page sizes. This PF supports a page size of 2^(n+12) if bit n is set. For example, if bit 0 is set, the Endpoint (EP) supports 4 KB 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.

14.4.4.9 PCIe SR-IOV System Page Size Register (0x180; RW)

Field	Bits	Init.	Type	Description
Page Size	31:0	0x1	RW	**System Page Size** This field defines the page size the system uses to map the VFs' memory addresses. Software must set the value of the *System Page Size* to one of the page sizes set in the *Supported Page Size* field. As with *Supported Page Sizes*, if bit n is set in *System Page Size*, the 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 *System Page Size*. The results are undefined if a bit is set in System Page Size that is not set in *Supported Page Size*. When *System Page Size* is set, the VFs are required to align all BAR resources on a *System Page Size* boundary. Each BAR size, including VF BARn Size (described later) must be aligned on a *System Page Size* boundary. Each BAR size, including VF BARn Size must be sized to consume a multiple of *System Page Size* bytes. All fields requiring page size alignment within a function must be aligned on a *System Page Size* boundary. *VF Enable* must be zero when *System Page Size* is set. The results are undefined if *System Page Size* is set when *VF Enable* is set.

14.4.4.10 PCIe SR-IOV BAR 0 - Low Register (0x184; RW)

Field	Bits	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. This bit is loaded from the NVM. It is reflected in the GLPCI_VFSUP register.
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 GLPCI_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.

14.4.4.11 PCIe SR-IOV BAR 0 - High Register (0x188; RW)

Field	Bits	Init.	Type	Description
BAR0 - MSB	31:0	0x0	RW	**BAR0 MSB** MSB part of BAR0.

14.4.4.12 PCIe SR-IOV BAR 2 Register (0x18C; RO)

Field	Bits	Init.	Type	Description
BAR2	31:0	0x0	RO	**BAR2** This BAR is not used.

14.4.4.13 PCIe SR-IOV BAR 3 - Low Register (0x190; RW)

Field	Bits	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. This bit is loaded from the NVM. It is reflected in the GLPCI_VFSUP register.
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 GLPCI_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.

14.4.4.14 PCIe SR-IOV BAR 3 - High Register (0x194; RW)

Field	Bits	Init.	Type	Description
BAR4 - MSB	31:0	0x0	RW	**BAR4 MSB** MSB part of BAR3.

14.4.4.15 PCIe SR-IOV BAR 5 Register (0x198; RO)

Field	Bits	Init.	Type	Description
BAR5	31:0	0x0	RO	**BAR5** This BAR is not used.

14.4.4.16 PCIe SR-IOV VF Migration State Array Offset Register (0x19C; RO)

Field	Bits	Init.	Type	Description
BIR	2:0	000b	RO	**BIR** Indicates which PF BAR contains the VF Migration State Array. Not implemented in the E810.
Offset	31:3	0x0	RO	**Offset** Offset, relative to the beginning of the BAR of the start of the migration array. Not implemented in the E810.

14.4.5 TPH Requester Capability

The TPH Requester capability is an optional extended capability to support TLP Processing Hints. The capability is exposed when the GLPCI_CAPSUP.*TPH_EN* bit is set from NVM.

Table 14-29 lists the TPH extended capability structure for PCIe devices.

Table 14-29. TPH Requester Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x1A0	PCI Express Extended Capability Header			
0x1A4	TPH Requester Capability Register			
0x1A8	TPH Requester Control Register			

Table 14-30 summarizes configuration sharing of the TPH Requester Capability registers among the different PCI functions.

Table 14-30. Configuration Sharing of the TPH Requester Capability

Field	Sub-field	Shared?	Replicated?	Comments	Section Reference
Enhanced Capability Header Register	Extended Capability ID	X			14.4.5.1
	Capability Version	X			
	Next Capability Offset		X		
TPH Requester Capability		X			14.4.5.2
TPH Requester Control			X		14.4.5.3
TPH ST Table			X	The Steering Table Upper fields are not supported.	

14.4.5.1 TPH Requester Extended Capability Header (0x1A0; RO)

Bits	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 14-20 for possible values of the next capability offset.

14.4.5.2 TPH Requester Capability Register (0x1A4; RO)

Bits	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 E810 does not support Interrupt Vector Mode of operation.
2	1b	RO	**Device Specific Mode** Set to indicate that the E810 supports Device Specific Mode of operation.
7:3	RsvdP	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	RsvdP	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 E810 does not support an ST Table
31:27	0x0	RsvdP	Reserved.

14.4.5.3 TPH Requester Control Register (0x1A8; RW)

Bits	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 E810) 010b = Device Specific Mode All other values are reserved. The default value of 000b indicates No Table mode of operation.
7:3	0x0	RsvdP	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 E810 is not permitted to issue transactions with TPH or Extended TPH as Requester. 01b = The E810 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 E810 is permitted to issue transactions with TPH and Extended TPH as Requester (the E810 does not issue transactions with Extended TPH). The default value of this field is 00b.
31:10	0x0	RsvdP	Reserved.

14.4.6 ACS Extended Capability Structure

The ACS Extended Capability defines a set of control points within a PCI Express topology to determine whether a TLP should be routed normally, blocked, or redirected. The capability is exposed when the GLPCI_CAPSUP.*ACS_EN* bit is set from NVM.

The ACS Capability structure is shared and exposed to all PFs.

Table 14-31 lists the PCIe ACS extended capability structure for PCIe devices.

Table 14-31. ACS Extended Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x1B0	PCI Express Extended Capability Header			
0x1B4	ACS Control Register (0x0)		ACS Capability Register (0x0)	

14.4.6.1 ACS Extended Capability Header (0x1B0; RO)

Bits	Init.	Type	Description
15:0	0x0D	RO	**PCI Express Extended Capability ID** PCIe extended capability ID indicating ACS capability.
19:16	0x1	RO	**Capability Version** PCIe ACS extended capability version number.
31:20	See description	RO	**Next Capability Offset** See Table 14-20 for possible values of the next capability offset.

14.4.6.2 ACS Capability Register (0x1B4; RO)

Bits	Init.	Type	Description
0	0b	RO	**ACS Source Validation (V)** Hard-wired to zero, not supported in the E810.
1	0b	RO	**ACS Translation Blocking (B)** Hard-wired to zero, not supported in the E810.
2	0b	RO	**ACS P2P Request Redirect (R)** Hard-wired to zero, not supported in the E810.
3	0b	RO	**ACS P2P Completion Redirect (C)** Hard-wired to zero, not supported in the E810.
4	0b	RO	**ACS Upstream Forwarding (U)** Hard-wired to zero, not supported in the E810.
5	0b	RO	**ACS P2P Egress Control (E)** Hard-wired to zero, not supported in the E810.
6	0b	RO	**ACS Direct Translated P2P (T)** Hard-wired to zero, not supported in the E810.
7	0b	RsvdP	Reserved.
15:8	0x0	RO	**Egress Control Vector Size** Hard-wired to zero, not supported in the E810.

14.4.6.3 ACS Control Register (0x1B6; RO)

Bits	Init.	Type	Description
0	0b	RO	**ACS Source Validation Enable (V)** Hard-wired to zero, not supported in the E810.
1	0b	RO	**ACS Translation Blocking Enable (B)** Hard-wired to zero, not supported in the E810.
2	0b	RO	**ACS P2P Request Redirect Enable (R)** Hard-wired to zero, not supported in the E810.
3	0b	RO	**ACS P2P Completion Redirect Enable (C)** Hard-wired to zero, not supported in the E810.
4	0b	RO	**ACS Upstream Forwarding Enable (U)** Hard-wired to zero, not supported in the E810.
5	0b	RO	**ACS P2P Egress Control Enable (E)** Hard-wired to zero, not supported in the E810.
6	0b	RO	**ACS Direct Translated P2P Enable (T)** Hard-wired to zero, not supported in the E810.
15:7	0x0	RsvdP	Reserved.

14.4.7 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 GLPCI_CAPSUP.*SEC_EN* bit is set from NVM.

Table 14-32 lists the Secondary PCI Express extended capability structure for PCIe devices.

Table 14-32. Secondary PCI Express Extended Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x1D0	PCI Express Extended Capability Header			
0x1D4	Link Control 3 Register			
0x1D8	Lane Error Status Register			
0x1DC	Equalization Control Register (Sized by Maximum Link Width)			
---	...			
0x1FA	...			

14.4.7.1 Secondary PCIe Extended Capability Header (0x1D0; RO)

Bits	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.

Bits	Init.	Type	Description
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 14-20 for possible values of the next capability offset.

14.4.7.2 Link Control 3 Register (0x1D4; RW)

Bits	Init.	Type	Description
0	0b	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 *Crosslink Supported* is 1b. This bit is not applicable and is RsvdP for Upstream Ports when the *Crosslink Supported* bit is 0b. The default value is 0b.
1	0b	RsvdP	**Link Equalization Request Interrupt Enable** When Set, this bit enables the generation of interrupt to indicate that the *Link Equalization Request* bit has been set. This bit is RW for Upstream Ports when *Crosslink Supported* is 1b. This bit is not applicable and is RsvdP for Upstream Ports when the *Crosslink Supported bit* is 0b. The default value for this bit is 0b.
8:2	0x0	RsvdP	Reserved.
15:9	0x0	RW	**Enable Lower SKP OS Generation Vector** When the Link is in L0 and the bit in this field corresponding to the current Link speed is set, SKP Ordered Sets are scheduled at the rate defined for SRNS, overriding the rate required based on the clock tolerance architecture. Bit definitions within this field are: Bit 0 = 2.5 GT/s Bit 1 = 5.0 GT/s Bit 2 = 8.0 GT/s Bit 3 = 16.0 GT/s Bits 6:4 = RsvdP
31:16	0x0	RsvdP	Reserved.

14.4.7.3 Lane Error Status Register (0x1D8; RW1CS)

The Lane Error Status register consists of a 32-bit vector, where each bit indicates if the corresponding PCI Express Lane detected an error.

Bits	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.

14.4.7.4 Lane Equalization Control Register (0x1DC - 0x1FA; RO)

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:

Bits	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.
7	0b	Rsvd	Reserved.
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.

14.4.8 Data Link Feature Extended Capability

Note: This feature should be enabled only in stand alone devices where there is a direct connection to Gen4 PCIe interface.

Table 14-33. Data Link Feature Extended Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x200	Data Link Feature - Extended Capability Header			
0x204	Data Link Feature - Capabilities Register			
0x208	Data Link Feature - Status Register			

14.4.8.1 Data Link Feature Extended Capability Header (0x200; RO)

Bits	Init.	Type	Description
15:0	0x0025	RO	**PCI Express Extended Capability ID** PCIe extended capability ID indicating Data Link feature capability.
19:16	0x1	RO	**Capability Version** PCIe Data Link feature extended capability version number.
31:20	See description	RO	**Next Capability Offset** See Table 14-20 for possible values of the next capability offset.

14.4.8.2 Data Link Feature Capabilities Register (0x204; RO)

Bits	Init.	Type	Description
22:0	0x1	RO	**Local Data Link Feature Supported** This field contains the Feature Supported value used when this Port sends a Data Link Feature DLLP. Defined features are: Bit 0 = Local Scaled Flow Control Supported This bit indicates that this Port supports the Scaled Flow Control Feature. Bits 22:1 = RsvdP
30:23	0x1	RO	Reserved.
31	1b	RO	**Data Link Feature Exchange Enable** If Set, this bit indicates that this Port will enter the DL_Feature negotiation state.

14.4.8.3 Data Link Feature Status Register (0x208; RO)

Bits	Init.	Type	Description
22:0	0x0	RO	**Remote Data Link Feature Supported** These bits indicate that the Remote Port supports the corresponding Data Link Feature. These bits capture all information from the Feature Supported field of the Data Link Feature DLLP even when this Port does not 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	0b	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	0x1	RO	Reserved.
31	1b	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

14.4.9 PASID Capability

Note: When PASID is enabled, SR-IOV is disabled (no support for concurrent PASID and SRIOV in the same system).

Note: PASID capability is not exposed in dummy functions.

Table 14-34. PASID Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x2D0	PASID Extended Capability Header			
0x2D4	PASID - Capabilities Register		PASID - Control Register	

14.4.9.1 PASID Extended Capability Header (0x2D0; RO)

Bits	Init.	Type	Description
15:0	0x1B	RO	**PCI Express Extended Capability ID** PCIe extended capability ID indicating PASID capability.
19:16	0x1	RO	**Capability Version** PCIe PASID extended capability version number.
31:20	See description	RO	**Next Capability Offset** See Table 14-20 for possible values of the next capability offset.

14.4.9.2 PASID Capabilities Register (0x2D4; RO)

Bits	Init.	Type	Description
0	0b	RsvdP	Reserved.
1	0b	RO	**Execute Permission Supported** 0b = The Endpoint never sets the *Execute Requested* bit. 1b = The Endpoint supports sending TLPs that have the *Execute Requested* bit set. Set to zero.
2	0b	RO	**Privileged Mode Supported** 0b = The Endpoint never sets the *Privileged Mode Requested* bit. 1b = The Endpoint supports operating in Privileged and Non-Privileged modes, and supports sending requests that have the *Privileged Mode Requested* bit set. Set to zero.
7:3	0x0	RsvdP	Reserved.
12:8	0x14 (20d)	RO	**Max PASID Width** Indicates the width of the PASID field supported by the Endpoint. The value n indicates support for PASID values 0 through 2n-1 (inclusive). The value 0 indicates support for a single PASID (0). The value 20 indicates support for all PASID values (20 bits). This field must be between 0 and 20 (inclusive).
15:13	0x0	RsvdP	Reserved.

14.4.9.3 PASID Control Register (0x2D6; RW)

Bits	Init.	Type	Description
0	0b	RW	**PASID Enable** 0b = The Endpoint is not permitted to do so. 1b = The Endpoint is permitted to send and receive TLPs that contain a PASID TLP Prefix.
1	0b	RsvdP	**Execute Permission Enable** Ignored, as *Execute Permission Supported* bit is cleared.
2	0b	RsvdP	**Privileged Mode Enable** Ignored, as *Privileged Mode Supported* bit is cleared.
15:3	0x0	RsvdP	Reserved.

14.4.10 Physical Layer 16.0 GT/s Capability

Table 14-35. Physical Layer 16.0 GT/s Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x210	Physical Layer 16.0 GT/s Extended Capability Header			
0x214	Physical Layer 16.0 GT/s - Capabilities Register			
0x218	Physical Layer 16.0 GT/s - Control Register			
0x21C	Physical Layer 16.0 GT/s - Status Register			
0x220	16.0 GT/s Local Data Parity Mismatch Status Register			
0x224	16.0 GT/s First Re-timer Data Parity Mismatch Status Register			
0x228	16.0 GT/s Second Re-timer Data Parity Mismatch Status Register			
0x22C	Spare			
16.0 GT/s Lane Equalization Control Registers				
0x230	Lane 3	Lane 2	Lane 1	Lane 0
0x234	Lane 7	Lane 6	Lane 5	Lane 4
0x238	Lane 11	Lane 10	Lane 9	Lane 8
0x23C	Lane 15	Lane 14	Lane 13	Lane 12

14.4.10.1 Physical Layer 16.0 GT/s Extended Capability Header (0x210; RO)

Bits	Init.	Type	Description
15:0	0x0026	RO	**PCI Express Extended Capability ID** PCIe extended capability ID indicating Physical Layer 16.0 GT/s Extended capability.
19:16	0x1	RO	**Capability Version** PCIe Data Link feature extended capability version number.
31:20	See description	RO	**Next Capability Offset** See Table 14-20 for possible values of the next capability offset.

14.4.10.2 Physical Layer 16.0 GT/s Capabilities Register (0x214; RsvdP)

Bits	Init.	Type	Description
31:0	0x0	RsvdP	Reserved.

14.4.10.3 Physical Layer 16.0 GT/s Control Register (0x218; RsvdP)

Bits	Init.	Type	Description
31:0	0x0	RsvdP	Reserved.

14.4.10.4 Physical Layer 16.0 GT/s Status Register (0x21C; RW1CS)

This register is valid only in function 0, and returns zero in other functions.

Bits	Init.	Type	Description
0	0b	ROS	**Equalization 16.0 GT/s Complete**
1	0b	ROS	**Equalization 16.0 GT/s Phase 1 Successful**
2	0b	ROS	**Equalization 16.0 GT/s Phase 2 Successful**
3	0b	ROS	**Equalization 16.0 GT/s Phase 3 Successful**
4	0b	RW1CS	**Link Equalization Request 16.0 GT/s**
31:5	0x0	RsvdZ	Reserved.

14.4.10.5 16.0 GT/s Local Data Parity Mismatch Status Register (0x220; RW1CS)

Bits	Init.	Type	Description
15:0	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.
31:16	0x0	RsvdZ	Reserved.

14.4.10.6 16.0 GT/s First Re-timer Data Parity Mismatch Status Register (0x224; RW1CS)

Bits	Init.	Type	Description
15:0	0x0	RW1CS	**First Re-timer 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.
31:16	0x0	RsvdZ	Reserved.

14.4.10.7 16.0 GT/s Second Re-timer Data Parity Mismatch Status Register (0x228; RW1CS)

Bits	Init.	Type	Description
15:0	0x0	RW1CS	**Second Re-timer 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.
31:16	0x0	RsvdZ	Reserved.

14.4.10.8 Physical Layer 16.0 GT/s Status 2 Register (0x22C; RsvdZ)

Bits	Init.	Type	Description
31:0	0x0	RsvdZ	Reserved.

14.4.10.9 16.0 GT/s Lane Equalization Control Register (0x230 - 0x23C; HWInit)

These registers define the control fields required for per-lane 16.0 GT/s equalization. A byte is allocated to each lane as follows:

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x230	Lane 3	Lane 2	Lane 1	Lane 0
0x234	Lane 7	Lane 6	Lane 5	Lane 4
0x238	Lane 11	Lane 10	Lane 9	Lane 8
0x23C	Lane 15	Lane 14	Lane 13	Lane 12

Each byte is defined as follows:

Bits	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**

14.4.11 Lane Margining at the Receiver Capability

Table 14-36. Lane Margining at the Receiver Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x250	Lane Margining at the Receiver Extended Capability Header			
0x254	Capabilities Register		Status Register	
0x258 - 0x294	Lane n - Status Register		Lane n - Control Register	

14.4.11.1 Lane Margining at the Receiver Capability Header (0x250; RO)

Bits	Init.	Type	Description
15:0	0x0027	RO	**PCI Express Extended Capability ID** PCIe extended capability ID indicating Lane Margining at the Receiver Capability.
19:16	0x1	RO	**Capability Version** PCIe Data Link feature extended capability version number.
31:20	See description	RO	**Next Capability Offset** See Table 14-20 for possible values of the next capability offset.

14.4.11.2 Lane Margining at the Receiver Capabilities Register (0x254; RO)

Bits	Init.	Type	Description
0	HwInit	RO	**Margining uses Driver Software**
15:1	0x0	RsvdP	Reserved.

14.4.11.3 Lane Margining at the Receiver Status Register (0x256; RO)

Bits	Init.	Type	Description
0	HwInit	RO	**Margining Ready**
1	HwInit	RO	**Margining Software Ready**
15:2	0x0	RsvdP	Reserved.

14.4.11.4 Margining Lane #n Control Register (0x258 + 4*n; RW)

Bits	Init.	Type	Description
2:0	000b	RW	**Receiver Number**
5:3	111b	RW	**Margin Type**
6	0b	RW	**Usage Model**
7	0b	RO	Reserved.
15:8	0x9C	RW	**Margin Payload**

14.4.11.5 Margining Lane #n Status Register (0x25A + 4*n; RO)

Bits	Init.	Type	Description
2:0	000b	RO	**Receiver Number Status**
5:3	000b	RO	**Margin Type Status**
6	0b	RO	**Usage Model Status**
7	0b	RO	Reserved.
15:8	0x0	RO	**Margin Payload Status**

14.5 Virtual Functions

14.5.1 Overview

14.5.1.1 VF to PF Allocation

The E810 supports up to 256 VFs. These VFs can be distributed arbitrarily among the different PFs. The distribution is done by NVM settings as the *TotalVFs* parameter should be stable at enumeration time.

For each of the potential Physical Functions (PFs), the following parameters are defined in the NVM:

- PCIe Configuration Space Control 1.*SR-IOV enable* — Should SR-IOV be exposed for this function?
- Per PF, First VF and Last VF — What is the first VF and the last VF allocated to each function (out of 256). Can be any number between 0 and 256-1.

From these parameters the following parameters are derived in the SR-IOV capability structure (see Section 14.4.4 for details):

- The SR-IOV structure is part of the configuration space of a PF only if the GLPCI_CAPSUP.*IOV_EN* bit is set in the NVM and PF_VT_PFALLOC.*VALID* field is set for this function.
- InitialVFs = TotalVFs = PF_VT_PFALLOC.*LASTVF*[n] - PF_VT_PFALLOC.*FIRSTVF* [n] +1
- First VF Offset = PF_VT_PFALLOC.*FIRSTVF* [n]+ 8 - PF# for ARI mode and PF_VT_PFALLOC.*FIRSTVF* [n]+ 256 - PF# for non ARI mode.

  Note: The First VF offset formula is defined so that the RID of a VF is fixed no matter which PF it belongs to.

- VF stride = 1

The First VF and last VF allocated to a PF can be read from the PF_VT_PFALLOC registers.

14.5.1.2 Bus-Device-Function Layout

The Requester ID allocation of the VF is done using the First VF Offset field and the VF stride in the IOV structure and is used to do the enumeration of the VFs.

14.5.1.2.1 ARI Mode

The ARI capability allows interpretation of the “Device” part of the Requester ID as part of the “Function” part.

The allocation of VFs to PF is flexible, there is no relationship between the PF RID and the associated VFs RID.

When the hierarchy is ARI-capable (VF ARI in SR-IOV Control register is set), the allocation is as follows:

Table 14-37. RID per VF - ARI Mode

VF#/PF#	B,D,F	Binary	Notes
PF 0	B,0,0	B,00000,000	
PF 1	B,0,1	B,00000,001	
PF 2	B,0,2	B,00000,010	
...	...	...	
PF 7	B,0,7	B,00000,111	
VF 0	B,1,0	B,00001,000	
VF 1	B,1,1	B,00001,001	
VF 2	B,1,2	B,00001,010	
...	...	...	
VF 247	B,31,7	B,11111,111	
VF 248	B+1.0.0	B+1,0000,000	
...	...	...	
VF 255	B+1,0,7	B+1,00000,111	

14.5.1.2.2 Non-ARI Mode

When the hierarchy is not ARI-capable (VF ARI in SR-IOV Control register is cleared), a non-zero PCI Device Number in the first bus can not be used. Thus, a second bus is needed to provide enough Requester IDs. In this mode the RID layout is as follows:

Table 14-38. RID per VF - non-ARI Mode

VF#/PF#	B,D,F	Binary	Notes
PF 0	B,0,0	B,00000,000	
PF 1	B,0,1	B,00000,001	
PF 2	B,0,2	B,00000,010	
...	...	...	
PF 7	B,0,7	B,00000,111	
VF 0	B+1,0,0	B+1,00000,000	
VF 1	B+1,0,1	B+1,00000,001	
VF 2	B+1,0,2	B+1,00000,010	
...	...	...	
VF 255	B+1,31,7	B+1,11111,111	

14.5.1.3 Configuration Space Overview

The configuration space reflected to each of the VF is a sparse version of the physical function configuration space. Table 14-39 describes the behavior of each register in the VF configuration space.

Table 14-39. 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 14.5.2.3.
	6	Status	Per VF	See Section 14.5.2.4.
	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 — Same value for all VFs of each PF	See Section 14.5.2.5.
	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	70	MSI-X Header	RO — 0xA011	Next = PCIe capability.
	72	MSI-X Message Control	Per VF	See Section 14.5.3.1.1.
	74	MSI-X Table Address	RO	See Section 14.5.3.1.2
	78	MSI-X PBA Address	RO	See Section 14.5.3.1.3

Table 14-39. VF PCIe Configuration Space [continued]

Section	Offset	Name	VF behavior	Notes
PCIe Capability	A0	PCIe Header	RO — 0x0010	Next = Last capability.
	A2	PCIe Capabilities	RO — as PF	
	A4	PCIe Dev Cap	RO — as PF	
	A8	PCIe Dev Ctrl	RW	As PF apart from FLR — See Section 14.5.3.2.1.
	AA	PCIe Dev Status	Per VF	See Section 14.5.3.2.2.
	AC	PCIe Link Cap	RO — as PF	
	B0	PCIe Link Ctrl	RO — 0x0	
	B2	PCIe Link Status	RO — 0x0	
	C4	PCIe Dev Cap 2	RO — as PF	
	C8	PCIe Dev Ctrl 2	RO — 0x0	
	D0	PCIe Link Ctrl 2	RO — 0x0	
	D2	PCIe Link Status 2	RO — 0x0	
AER Capability	100	AER — Header	RO — 0x15010002	Next = ARI structure.
	104	AER — Uncorr Status	Per VF	See Section 14.5.3.3.1.
	108	AER — Uncorr Mask	RO — 0x0	
	10C	AER — Uncorr Severity	RO — 0x0	
	110	AER — Corr Status	Per VF	See Section 14.5.3.3.2.
	114	AER — Corr Mask	RO — 0x0	
	118	AER — Cap/Ctrl	RO	See Section 14.5.3.3.3.
	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	150	ARI — Header	0x1A01000E	Next = TPH Structure.
	154	ARI — Cap/Ctrl	RO — 0X0	
TPH Requester Capability	1A0	TPH - Header	0x1D010017	Next = ACS Structure.
	1A4	TPH - Capability	RO — 0x00000005	No table reported..
	1A8	TPH - Control	Per VF	Same structure as in PF.
ACS Capability	1D0	ACS - Header	RO — 0x0001000D	Next = Last extended capability.
	1D4	ACS - Capability	RO — 0x00000000	

14.5.2 Mandatory Configuration Space

The IOV specification defines the configuration space of the Virtual functions as a mirror of the Physical function configuration space with the exception of some fields that are implemented per VF.

This section describes the expected handling of the different part of the configuration space for virtual functions. It deals only with the parts relevant to the E810 and describes only changes that are not a trivial implementation of the specification.

14.5.2.1 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 to the whole set of VFs.

All the legacy error reporting bits are emulated for the VF.

14.5.2.2 Memory BARs Assignment

The 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 the 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, thus only those are replicated.

Table 14-40 describes the BARs and the stride used for the VFs:

Table 14-40. VF BARs in the E810

BAR	Type	Usage	Requested Size per VF
0	Mem	CSR space	See Section 13.1.2.2
1	Mem	High word of CSR space address	N/A
2	N/A	Not used	N/A
3	Mem	MSI-X	max (16K, page size)
4	Mem	High word of MSI-X space address	N/A
5	N/A	Not used	N/A

14.5.2.3 VF Command Register (0x4; RW)

Bits	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 *Bus Master Enable* set must not be blocked by transactions for VFs that have their *Bus Master Enable* 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.

Bits	Init.	Type	Description
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.

14.5.2.4 VF Status Register (0x6; RW)

Bits	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 E810 VFs implement extended capabilities. E810 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.
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.

14.5.2.5 VF Subsystem ID (0x2E; RO)

This value is loaded from NVM if the GLPCI_CAPSUP.*LOAD_SUBSYS_ID* bit is set. Each VF is loaded from the respective PF's NVM PFPCI_SUBSYSID.*VF_SUB_ID* field (in other words, all VFs of a specific PF share the same value).

14.5.3 PCI and PCIe Capabilities

The following capability structures are partially replicated in VFs configuration space:

- PCIe capability structure
- MSI-X capability structure

The following extended capability structures are partially replicated in VFs config space:

Table 14-41. Extended Capabilities List

Address Range	Item	Cases Where Capability Does Not Exist	Next Pointer
0x100 - 0x128	Advanced Error Reporting (AER)	None (always present)	Any of the below / 0x000
0x148 - 0x14C	Alternative RID Interpretation (ARI)	ARI Enabled bit in NVM is set to 0b	Any of the below / 0x000
0x1A0 - 0x1A8	TPH Requester	TPH Enabled bit in NVM is set to 0b	Any of the below / 0x000
0x1B0 - 0x1B4	Access Control Services (ACS)	ACS Enabled bit in NVM is set to 0b	0x000

14.5.3.1 MSI-X Capability

The MSI-X BAR size is max(16K, page size).

The location and size of the MSI-X vector table and the MSI-X Pending Bits table are determined as follows:

- MSI-X vector table
  - The MSI-X table structure (Section 14.3.3.2) 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.
  - Starts at offset 0x0000 from start of BAR.
  - Contains the MSI-X vectors for the VF. The number of entries in the table (N) is set from NVM. The maximum value of N is 65.
  - The vectors start with the “Vector 0” (one per VF), followed by the other vectors allocated to the VF.
- MSI-X Pending Bits table
  - The PBA structure (Section 14.3.3.2.2) 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.
  - Starts at half the BAR size (default is offset 0x2000 - 8 KB from start of BAR).
  - Contains the pending bits for the VF. The VF is allocated one 64-bit register for a maximum of 17 bits.
  - The bits start with the “Vector 0” bit (one per VF), followed by bits for the other vectors allocated to the VF.

14.5.3.1.1 VF MSI-X Control Register (0x72; RW)

Bits	Init.	Type	Description
10:0	0x40 (65 vectors)	RO	**TS** Table Size (N-1). N varies with the number of virtual functions as set from NVM. This field is loaded from NVM. It is generated from the VPINT_ALLOC[VF#].*LAST* - VPINT_ALLOC[VF#].*FIRST* fields.
13:11	000b	RO	Reserved.
14	0b	RW	**Mask** Function Mask.
15	0b	RW	**En** MSI-X Enable.

14.5.3.1.2 MSI-X Address Register (0x74; RO)

Bits	Init.	Type	Description
2:0	011b	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.
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 vectors address. The lower three *Table BIR* bits are masked off (set to 0b) by software to form a 32-bit QWord-aligned offset.

14.5.3.1.3 MSI-X PBA Register (0x78; RO)

Bits	Init.	Type	Description
2:0	0x3	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. A BIR value of three indicates that the PBA is mapped in BAR 3.
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 *PBA BIR* 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 requested BAR size.

14.5.3.2 PCIe Capability Registers

The Device Control and Device Status registers have some fields that are specific per VF.

14.5.3.2.1 VF Device Control Register (0xA8; RW)

Bits	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.
8	0b	RO	**Extended Tag field Enable**
9	0b	RO	**Phantom Functions Enable** Not implemented in the E810.
10	0b	RO	**Auxiliary Power PM Enable** Zero for VFs.
11	0b	RO	**Enable No Snoop** Zero for VFs.
14:12	000b	RO	**Max Read Request Size** Zero for VFs.
15	0b	RW	**Initiate Function Level Reset** Specific to each VF.

14.5.3.2.2 VF Device Status Register (0xAA; RO)

Bits	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 E810 received an unsupported request. This field is separate per VF. However, in the case where an error cannot be associated with a VF, this bit is set in all PFs and VFs.
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.

14.5.3.3 AER Registers

The following registers in the AER capability have a different behavior in a VF function.

Note: Unlike the PF AER registers, these registers are not sticky since the VF is reset on FLR and on in-band reset.

14.5.3.3.1 Uncorrectable Error Status Register (0x104; RW1C)

Bits	Init.	Type	Description
3:0	0x0	RO	Reserved.
4	0b	RO	**Data Link Protocol Error Status** Hard-wired to 0b.
5	0b	RO	**Surprise Down Error Status** Hard-wired to 0b.
11:6	0x0	RO	Reserved.
12	0b	RW1C	**Poisoned TLP Status**
13	0b	RO	**Flow Control Protocol Error Status** Hard-wired to 0b
14	0b	RW1C	**Completion Timeout Status**
15	0b	RW1C	**Completer Abort Status**
16	0b	RW1C	**Unexpected Completion Status**
17	0b	RO	**Receiver Overflow Status** Hard-wired to 0b
18	0b	RO	**Malformed TLP Status** Hard-wired to 0b
19	0b	RO	**ECRC Error Status** Hard-wired to 0b
20	0b	RW1C	**Unsupported Request Error Status** When caused by a function that claims a TLP.
21	0b	RO	**ACS Violation Status** Hard-wired to 0b
31:22	0x0	RO	Reserved.

14.5.3.3.2 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.

Bits	Init.	Type	Description
0	0b	RO	**Receiver Error Status** Hard-wired to 0b
5:1	0x0	RO	Reserved.
6	0b	RO	**Bad TLP Status** Hard-wired to 0b
7	0b	RO	**Bad DLLP Status** Hard-wired to 0b
8	0b	RO	**REPLAY_NUM Rollover Status** Hard-wired to 0b
11:9	000b	RO	Reserved.
12	0b	RO	**Replay Timer Timeout Status** Hard-wired to 0b
13	0b	RW1C	**Advisory non-Fatal Error Status**
31:14	0x0	RO	Reserved.

14.5.3.3.3 Advanced Error Capabilities and Control Register (0x118; RO)

Bits	Init.	Type	Description
4:0	0x0	ROS	**First Recorded Error** 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	0b	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 GLPCI_CAPSUP register
6	0b	RO	**ECRC Generation Enable** When set, ECRC generation is enabled. Hard-wired to 0b. The PF setting applies to the VF.
7	0b	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 GLPCI_CAPSUP register.
8	0b	RO	**ECRC Check Enable** When set, ECRC checking is enabled. Hard-wired to 0b. The PF setting applies to the VF.
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	RO	Reserved.

NOTE: ***This page intentionally left blank.***

Chapter 15 Reliability, Diagnostics, and Testability

15.1 Reliability

15.1.1 ECC Support and ECC Error Flow

Memories in the E810 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 E810 reacts to such an event.

Reporting of ECC errors is as follows:

- The *_ECC_COR_ERR and *_ECC_UNCOR_ERR per-block registers count the occurrence of correctable and uncorrectable errors, respectively (the * symbol stands for the block name).

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 might reach host memory.
- Recovery then depends on the memory where the error happens:
  - An error takes place in the core or global domains.
    - If the GLGEN_RSTCTL.*ECC_RST_ENA* bit is set to 1b, the E810 generates a GLOBR reset.
  - An error takes place in the EMP.
    - If the GLMNG_WD_ENA.*ECC_RST_ENA* bit is set to 1b, the EMP generates an EMPR reset.
  - An error takes place in FLEEP (Shadow RAM).
    - An ECC error check in the Shadow RAM is enabled via the *Shadow RAM ECC Enable* bit in the NVM.
    - The FLEEP reloads the Shadow RAM from the NVM.
    - If the GLGEN_RSTCTL.*ECC_RST_ENA* bit is set to 1b, the E810 generates a GLOBR 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 the GLMNG_WD_ENA.ECC_RST_ENA bit is set to 1b, the EMP generates an EMPR reset.
    - Else, if the GLGEN_RSTCTL.ECC_RST_EN bit is set to 1b, the E810 generates a GLOBR reset.

15.2 Link Loopback Operations

Loopback operations are supported by the E810 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 E810 and the system (such as PCIe bus connections, and so on).

15.3 Firmware Recovery Mode

15.3.1 Overview

Firmware 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.

15.3.1.1 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.
- CORER/GLOBR firmware flow failure.
- Repeated Watchdog reset.
- Repeated Memory exception.

15.3.1.2 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.

15.3.2 Recovery Flows

When firmware detects a failure scenario, it first attempts to automatically rollback to the non-active NVM bank (see Section 15.3.2.1).

If rollback fails or the non-active bank is not valid, firmware attempts recovery (see Section 15.3.2.2). 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).

15.3.2.1 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.

15.3.2.2 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.

15.3.2.2.1 Recovery Firmware Supported Features

15.3.2.2.1.1 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 9.5.12).

- Section 9.5.13.1, "Get Version (0x0001)"
- Section 9.5.13.4, "Set PF Context (0x0004)"
- Section 9.5.13.5, "Get Expanded AQ Error Reason (0x0005)"
- Section 9.5.13.6, "Request Resource Ownership (0x0008)"
- Section 9.5.13.7, "Release Resource Ownership (0x0009)"

- Section 3.4.10.1, "NVM Read (0x0701)"
- Section 3.4.10.2, "NVM Erase (0x0702)"
- Section 3.4.10.3, "NVM Write (0x0703)"
- Section 3.4.10.8, "NVM Write Activate (0x0707)"

15.3.2.2.1.2 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.

15.3.2.2.2 Reset Flows

15.3.2.2.2.1 CORER/GLOBR

Recovery firmware performs CORER/GLOBR with no auto-load or initializations.

Firmware acknowledges to software that reset is done and AL is done, as it does in operational firmware.

15.3.2.2.2.2 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.

15.3.2.2.2.3 PCIR/PERST

Recovery firmware performs minimal configuration to enable AQ communications.

Recovery firmware auto-loads 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 acknowledges to software that the 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.

15.3.2.2.2.4 POR/EMPR

Recovery firmware auto-loads 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.

15.3.2.2.3 Unsupported Features

Recovery firmware does not support:

- Link
- Traffic
- Manageability (BMC)

Recovery firmware only supports firmware update flows from host.

15.3.3 Operation Mode Software Identification

Firmware indicates operation mode via the *FW_MODES* field (Bits 2:0) in the GL_MNG_FWSM CSR (see Section 13.2.2.29.7). 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.

NOTE: ***This page intentionally left blank.***

Chapter 16 Electrical/Mechanical Specification

16.1 Introduction

This section describes electrical and mechanical characteristics of the E810, including: Operating Conditions, Power Delivery, Power Dissipation, DC/AC Specifications, Package, and Supported Devices.

16.2 Operating Conditions

16.2.1 Absolute Maximum Ratings

Table 16-1. Absolute Maximum Ratings

Symbol	Parameter	Min	Max	Units
T_j	Junction Temperature Under Bias	-0	115	°C
$T_{storage}$	Storage Temperature Range	-40	115	°C
Vi	3.3V I/O Input Voltage	VSS-0.3	VDDIO33+0.3	V
VDDIO33	3.3V Digital I/O Supply Voltage	VSS-0.3	3.645	V

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 Tj of 125 °C is allowed up to 4% of the time without affecting device reliability.

16.2.2 Recommended Operating Conditions

Table 16-2. Recommended Operating Conditions

Symbol	Parameter	Min	Typical	Max	Units
Tj	Junction Temperature	0		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.
- Refer to Chapter 18, "Thermal Design Considerations" for a description of the allowable thermal environment.
- Thermal design power (TDP) is specified at 105 °C junction temperature.

16.3 Power Delivery

16.3.1 Power Supply Specification

Table 16-3. 3.3 V Power Supply Specification

VDDIO33 (3.3V) Parameters				
Title	**Description**	**Min**	**Max**	**Units**
Rise Time	Time from 10% to 90% mark.	0.1	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*V (min)/rise time (max) Max: 0.8*V (max)/rise time (min)	24	28,800	V/s
Operational Range	Voltage range for normal operating conditions (+5%, -10%).	2.97	3.465	V
Ripple	Maximum voltage ripple (peak-to-peak).	N/A	VDDIO33 ±200	mV
Overshoot	Maximum overshoot allowed.	N/A	100	mV
Overshoot Settling Time	Maximum overshoot allowed duration. (At that time delta voltage should be lower than 5 mV from steady state voltage.)	N/A	0.05	ms

Note: If ViH tolerance to 3.6 V is required, maintain VDDIO33 between 3.3 V to 3.465 V (3.38 ±2.5%).

Table 16-4. 1.8 V VDDH Power Supply Specification

VDDH Parameters				
Title	**Description**	**Min**	**Max**	**Units**
Rise Time	Time from 10% to 90% mark.	0.1	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*V (min)/rise time (max) Max: 0.8*V (max)/rise time (min)	0.33	7120	V/s
Operational Range	Voltage range for normal operating conditions (±10%).	1.62	1.98	V
Ripple	Maximum voltage ripple (peak-to-peak).	N/A	20	mV
Overshoot	Maximum overshoot allowed.	N/A	50	mV
Overshoot Duration	Maximum overshoot allowed duration. (At that time delta voltage should be lower than 5 mV from steady state voltage.)	0	0.05	ms

Table 16-5. 1.1 V AVDDH Power Supply Specification

AVDDH Parameters				
Title	**Description**	**Min**	**Max**	**Units**
Rise Time	Time from 10% to 90% mark.	0.1	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*V (min)/rise time (max) Max: 0.8*V (max)/rise time (min)	0.33	7120	V/S
Operational Range	Voltage range for normal operating conditions (±5%)/	1.045	1.155	V
Ripple	Maximum voltage ripple (peak-to-peak). 600 KHz to 20 MHz	N/A	10	mV
Overshoot	Maximum overshoot allowed.	N/A	50	mV
Overshoot Duration	Maximum overshoot allowed duration. (At that time delta voltage should be lower than 5 mV from steady state voltage.)	0	0.05	ms

Table 16-6. 0.9 V AVDD_ETH and AVDD_PCIE Power Supply Specification

AVDD (AVDD_PCIE/AVDD_ETH) Parameters				
Title	**Description**	**Min**	**Max**	**Units**
Rise Time	Time from 10% to 90% mark.	0.1	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*V (min)/rise time (max) Max: 0.8*V (max)/rise time (min)	0.33	7120	V/S
Operational Range	Voltage range for normal operating conditions (±5%).	0.855	0.945	V
Ripple	Maximum voltage ripple (peak to peak). 600 KHz to 20 MHz	N/A	2.0	mV
Overshoot	Maximum overshoot allowed.	N/A	50	mV
Overshoot Duration	Maximum overshoot allowed duration. (At that time delta voltage should be lower than 5 mV from steady state voltage.)	0	0.05	ms

Note: In the E810-CAM2/CAM1, AVDD is split for the PCIe and Ethernet interfaces. In the E810-XXVAM2, a single AVDD supplies both interfaces.

Table 16-7. 0.8 V VDD Power Supply Specification

VDD Parameters				
Title	**Description**	**Min**	**Max**	**Units**
Rise Time	Time from 10% to 90% mark.	0.1	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*V (min)/rise time (max) Max: 0.8*V (max)/rise time (min)	0.33	7120	V/S
Operational Range	Voltage range for normal operating conditions (0.8 V ±5%).	0.76	0.84	V
Ripple	Maximum voltage ripple (peak to peak).	N/A	20	mV
Overshoot	Maximum overshoot allowed.	N/A	50	mV
Overshoot Duration	Maximum overshoot allowed duration. (At that time delta voltage should be lower than 5 mV from steady state voltage.)	0	0.05	ms

16.3.1.1 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:

- The E810 provides an internal Power-On Reset (POR). An alternate external POR (LAN_POWER_GOOD) can be used to extend the reset if necessary.
- The E810 initialization budgets 35 ms from system power-on to LAN_POWER_GOOD (or internal POR) completion. Remaining 65 ms (for the PCIe spec 100 ms system power to PERST# inactive) is allocated to internal initialization.
- The internal POR function monitors VDDH and VDD only. Internal POR signal is held active for 7-10 ms after these supplies reach ~50%.
- Application of power should begin with the 3.3 V (VDDIO33) and 1.8 V (VDDH) supplies ***at the same time***. After 3.3 V reaches 80% of its final value, the digital core 0.8 V supply (VDD) should be enabled and reach its final value within 10 ms.
- 156.25 MHz REFCLK oscillator should be powered with same VDDIO33 used for the E810, but the oscillator's output enable should be held inactive until VDDH is on. Suggest asserting REFCLK OE at the same time as VDD regulator is enabled. This sequence avoids driving a signal into the un-powered clock buffer while enabling REFCLK prior to any analog supply. Allow 10 ms oscillator startup time from 3.3 V to analog supplies enable.
- AVDD_ETH, AVDD_PLL, and REFCLK should be active within 7 ms of VDD regulator enabling to insure that internal clock is active prior to internal POR complete. If this cannot be met, the LAN_POWER_GOOD signal should be used to extend reset. The 35 ms budget must still be met for PCIe timing.
- Use slow rise time on AVDD_ETH to minimize power-on current spike (~3 ms). This allows internal clock and reset to propagate before AVDD has reached its final value.
- AVDD_PCIE and AVDDH can be enabled with AVDD_ETH, or after a slight delay. A small delay (1-3 ms) will mitigate turn on current spikes from these supplies.
- AVDD_PLL should always be sequenced with AVDD_ETH.

Figure 16-1 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.

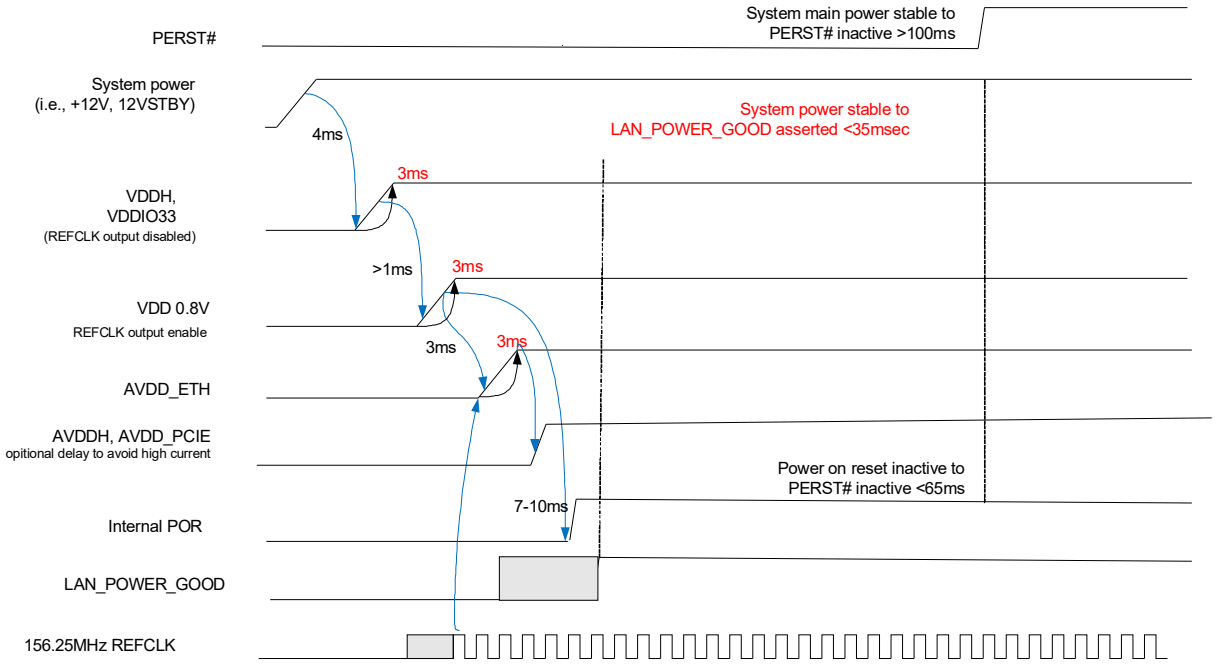


Figure 16-1. Power Supply Sequence

16.3.2 In-Rush Current

Max Load Step (A):

- AVDD = 0.9
- VDD = 5.0

Max Slew Rate (A/µs):

- AVDD = 0.2
- VDD = 2.0

16.4 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 consumption is listed separately for the E810-CAM2/CAM1 versus the E810-XXVAM2.

Power numbers are provides 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 Tj = 80 °C.

16.4.1 Max Power (TDP) - E810-CAM2/CAM1

Table 16-8. E810-CAM2/CAM1 MAX Power - PCIe Gen 4 and PCIe Gen 3

	Link Speed					
	2x100G PCIe Gen4x16	1x100G/ 2x50G/4x25G PCIe Gen4x16	2x25G PCIe Gen4x8	2x100G PCIe Gen3x16	1x100G/ 2x50G/4x25G PCIe Gen3x16	2x25G PCIe Gen3x8
VDDIO33 (A)	0.02	0.02	0.02	0.02	0.02	0.02
VDDH (A)	0.02	0.02	0.02	0.02	0.02	0.02
AVDDH (A)	0.80	0.50	0.20	0.80	0.50	0.20
AVDD_ETH (A)	3.00	1.80	0.90	3.00	1.80	0.90
AVDD_PCIE (A)	3.904	3.90	2.00	2.60	2.60	1.40
VDD (A)	10.60	10.00	8.80	10.30	9.70	8.50
Power (W)	15.67	13.78	9.97	14.26	12.37	9.19

Notes:

- VCCIO33 current includes on-chip power dissipation only for TDP calculation. System power supply should account for external 3.3 V powered devices, such as LEDs, flash, oscillator, and pluggable modules.
- AVDD_ETH current includes 0.02A for the AVDD_PLL supply.

16.4.2 Typical Power - E810-CAM2/CAM1

Table 16-9. E810-CAM2/CAM1 Typical Active Power - PCIe Gen 4 and PCIe Gen 3

	Link Speed					
	2x100G PCIe Gen4x16	1x100G/ 2x50G/4x25G PCIe Gen4x16	2x25G PCIe Gen4x8	2x100G PCIe Gen3x16	1x100G/ 2x50G/4x25G PCIe Gen3x16	2x25G PCIe Gen3x8
VDDIO33 (A)	0.02	0.02	0.02	0.02	0.02	0.02
VDDH (A)	0.02	0.02	0.02	0.02	0.02	0.02
AVDDH (A)	0.70	0.50	0.20	0.70	0.50	0.20
AVDD_ETH (A)	2.80	1.60	0.70	2.70	1.50	0.80
AVDD_PCIE (A)	3.70	3.70	1.80	2.30	2.20	1.20
VDD (A)	7.60	7.00	6.10	7.00	6.40	5.60
Power (W)	12.80	11.00	7.45	11.00	9.10	6.60

Note: Typical conditions: typical material, TJ = 80 °C, nominal voltages. and continuous network traffic at link speed.

Table 16-10. E810-CAM2/CAM1 Typical Idle Power - PCIe Gen 4 and PCIe Gen 3

	Link Speed					
	2x100G PCIe Gen4x16	1x100G/ 2x50G/4x25G PCIe Gen4x16	2x25G PCIe Gen4x8	2x100G PCIe Gen3x16	1x100G/ 2x50G/4x25G PCIe Gen3x16	2x25G PCIe Gen3x8
VDDIO33 (A)	0.02	0.02	0.02	0.02	0.02	0.02
VDDH (A)	0.02	0.02	0.02	0.02	0.02	0.02
AVDDH (A)	0.70	0.50	0.20	0.70	0.50	0.20
AVDD_ETH (A)	2.80	1.60	0.70	2.70	1.50	0.80
AVDD_PCIE (A)	3.70	3.70	1.80	2.30	2.20	1.20
VDD (A)	6.40	6.00	5.30	6.10	5.60	5.00
Power (W)	11.84	10.22	6.81	10.25	8.46	6.12

Note: Typical conditions: typical material, TJ = 80 °C, nominal voltages. and no traffic.

Table 16-11. E810-CAM2/CAM1 Typical D3 with Wake-Up Enabled - Tj = 80 °C

	Link Speed		
	2x100G	1x100G/2x50G/4x25G	2x25G
VDDIO33 (A)	0.02	0.02	0.02
VDDH (A)	0.02	0.02	0.02
AVDDH (A)	0.70	0.50	0.20
AVDD_ETH (A)	2.80	1.60	0.70
AVDD_PCIE (A)	0.10	0.10	0.10
VDD (A)	6.40	6.00	5.30
Power (W)	8.60	7.00	5.28

Table 16-12. E810-CAM2/CAM1 Typical D3 with Wake-Up Enabled - Tj = 25 °C

	Link Speed		
	2x100G	1x100G/2x50G/4x25G	2x25G
VDDIO33 (A)	0.02	0.02	0.02
VDDH (A)	0.02	0.02	0.02
AVDDH (A)	0.70	0.50	0.20
AVDD_ETH (A)	3.00	1.70	0.80
AVDD_PCIE (A)	0.10	0.10	0.10
VDD (A)	6.00	5.40	4.90
Power (W)	8.46	6.60	5.00

Table 16-13. E810-CAM2/CAM1 Typical D3 with Wake-Up Disabled - Tj = 25 °C

	Link Speed		
	2x100G	1x100G/2x50G/4x25G	2x25G
VDDIO33 (A)	0.02	0.02	0.02
VDDH (A)	0.02	0.02	0.02
AVDDH (A)	0.70	0.50	0.20
AVDD_ETH (A)	0.40	0.40	0.40
AVDD_PCIE (A)	0.10	0.10	0.10
VDD (A)	6.00	5.40	4.90
Power (W)	6.10	5.40	4.70

16.4.3 Max Power (TDP) - E810-XXVAM2

Table 16-14. E810-XXVAM2 MAX Power - PCIe Gen 4 and PCIe Gen 3

	Link Speed					
	2x25G PCIe Gen4x8	1x25G PCIe Gen4x8	1x10G PCIe Gen4x8	2x25G PCIe Gen3x8	1x25G PCIe Gen3x8	1x10G PCIe Gen3x8
VDDIO33 (A)	0.02	0.02	0.02	0.02	0.02	0.02
VDDH (A)	0.02	0.02	0.02	0.02	0.02	0.02
AVDD (A)	2.7	2.5	2.4	2.1	1.9	1.8
VDD (A)	10.0	9.6	9.3	9.7	9.3	9.0
Power (W)	10.53	10.03	9.7	9.75	9.25	8.92

16.4.4 Typical Power - E810-XXVAM2

Table 16-15. E810-XXVAM2 Typical Active Power - PCIe Gen 4 and PCIe Gen 3

	Link Speed					
	2x25G PCIe Gen4x8	1x25G PCIe Gen4x8	1x10G PCIe Gen4x8	2x25G PCIe Gen3x8	1x25G PCIe Gen3x8	1x10G PCIe Gen3x8
VDDIO33 (A)	0.02	0.02	0.02	0.02	0.02	0.02
VDDH (A)	0.02	0.02	0.02	0.02	0.02	0.02
AVDD (A)	2.4	2.1	2.0	1.9	1.6	1.6
VDD (A)	6.1	5.6	5.3	5.8	5.5	5.1
Power (W)	7.1	6.5	6.1	6.5	5.9	5.6

Note: Typical conditions: typical material, TJ = 80 °C, nominal voltages. and continuous network traffic at link speed.

Table 16-16. E810-XXVAM2 Typical Idle Power - PCIe Gen 4 and PCIe Gen 3

	Link Speed					
	2x25G PCIe Gen4x8	1x25G PCIe Gen4x8	1x10G PCIe Gen4x8	2x25G PCIe Gen3x8	1x25G PCIe Gen3x8	1x10G PCIe Gen3x8
VDDIO33 (A)	0.02	0.02	0.02	0.02	0.02	0.02
VDDH (A)	0.02	0.02	0.02	0.02	0.02	0.02
AVDD (A)	2.4	2.1	2.0	1.9	1.6	1.6
VDD (A)	5.3	4.8	4.5	5.0	4.7	4.3
Power (W)	6.5	5.8	5.5	5.8	5.3	5.0

Note: Typical conditions: typical material, TJ = 80 °C, nominal voltages. and no traffic.

Table 16-17. E810-XXVAM2 Typical D3 with Wake-Up Enabled - Tj = 25 °C

	Link Speed		
	2x25G	**2x10G**	**2x1G**
VDDIO33 (A)	0.02	0.02	0.02
VDDH (A)	0.02	0.02	0.02
AVDD (A)	0.7	0.4	0.2
VDD (A)	4.5	4.3	4.2
Power (W)	4.3	4.9	3.6

Table 16-18. E810-XXVAM2 Typical D3 with Wake-Up Disabled - Tj = 25 °C

	Link Speed		
	2x25G	**2x10G**	**2x1G**
VDDIO33 (A)	0.02	0.02	0.02
VDDH (A)	0.02	0.02	0.02
AVDD (A)	0.05	0.05	0.05
VDD (A)	4.2	4.2	4.2
Power (W)	3.5	3.5	3.5

16.5 DC/AC Specification

16.5.1 Digital I/O DC Specifications

Table 16-19. 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	VDDIO33+300 mV	V
VIL	Input Low Voltage		-0.3	0.8	V
Iil	Input Current[1]	VI=3.3 V / 0V		10	µA
PU	Internal pull-up		10	20	KΩ

1. Input Leakage Current

16.5.1.1 Open Drain I/O DC Specification

This section applies to SMBD, SMBCLK, SMBALRT _N, PE_WAKE_N.

Table 16-20. Open Drain I/O DC Characteristics

Symbol	Parameter	Condition	Min	Max	Units
Vih	Input High Voltage		2.0	VDDIO33 +0.3	V
Vil	Input Low Voltage		-0.2	0.8	V
Ileakage	Output Leakage Current[1]	0 ≤ Vin ≤ VDDIO33 max		100	µA
Vol	Output Low Voltage	@ Ipullup = 4 mA	0.4		V
Cin	Input Pin Capacitance				pF
Ioffsmb	Input leakage Current[1]	VDDIO33 off or floating		100	µA
IOL	Output Current Low		6		mA

1. Device meets this specification whether powered or un-powered.

16.5.1.2 NC-SI I/O DC Specification

Table 16-21. 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	Iol = 4 mA, Vref = $Vref_{min}$	0		0.4	V
Voh	Output High Voltage	Iol = -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 Vrefmax up to 3.6 V, the E810 supports the Vrefmax up to 3.46 V (3.3 V +5%).

16.5.2 Digital I/F AC Specifications

16.5.2.1 Digital I/O AC Specifications

Table 16-22. 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	

Note: Does not apply to open drain outputs.

16.5.2.2 SMBus and I²C AC Specifications

The E810 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 E810 also supports a 400 KHz SMBus and I²C (as a slave), and meets the specifications listed in Table 16-23:

Table 16-23. 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

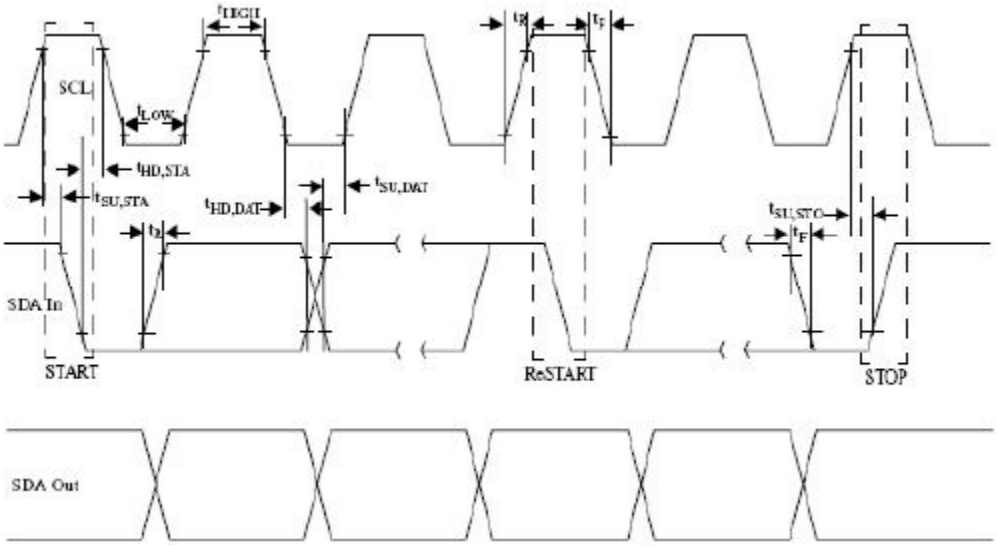


Figure 16-2. SMBus I/F Timing Diagram

16.5.2.3 FLASH AC Specification

The E810 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 16-24 and Figure 16-3.

Note: Table 16-24 applies to FLSH_SI, FLSH_SO, FLSH_SCK and FLSH_CE_N.

Table 16-24. 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 VDDIO33/2.

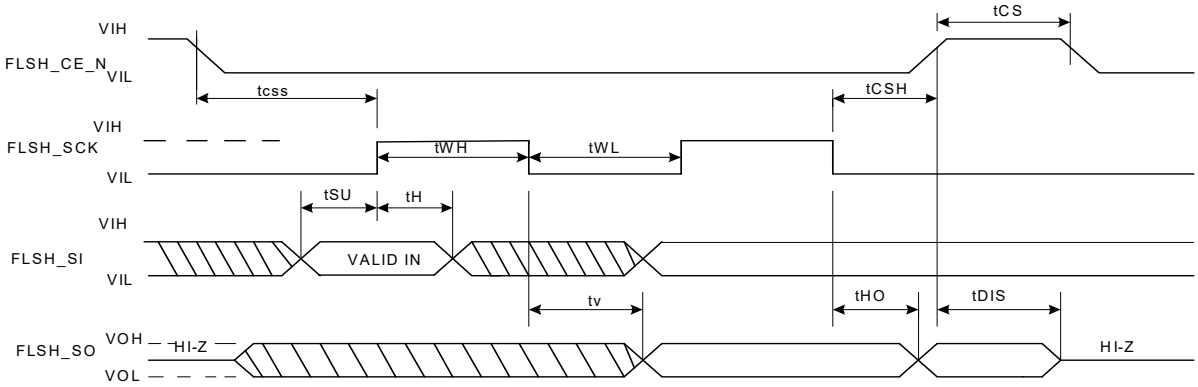


Figure 16-3. Flash I/F Timing Diagram

16.5.2.4 NC-SI AC Specifications

The E810 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 16-25 and Figure 16-4.

Table 16-25. 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 pF ≤ C_{load} ≤ 50 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			µs
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 16-21). 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.

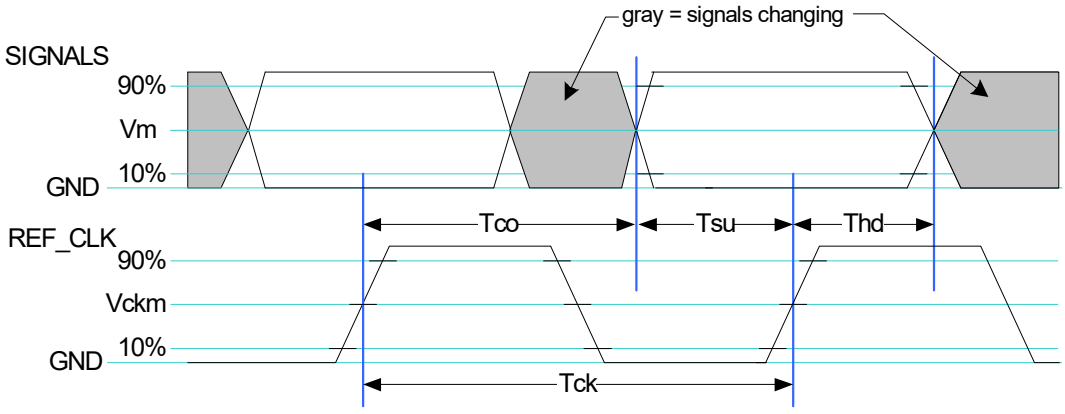


Figure 16-4. NC-SI AC Timing Diagram

16.5.2.5 JTAG AC Specification

The E810 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 16-26 and Figure 16-5.

Table 16-26. JTAG I/F Timing Parameters

Symbol	Parameter	Min	Typical	Max	Units
t_{JCLK}	JTCK Clock Frequency			10	MHz
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

Notes:

- Table 16-26 applies to JTCK, JTMS, JTDI and JTDO.
- Timing measured relative to JTCK reference voltage of VDDIO33/2.

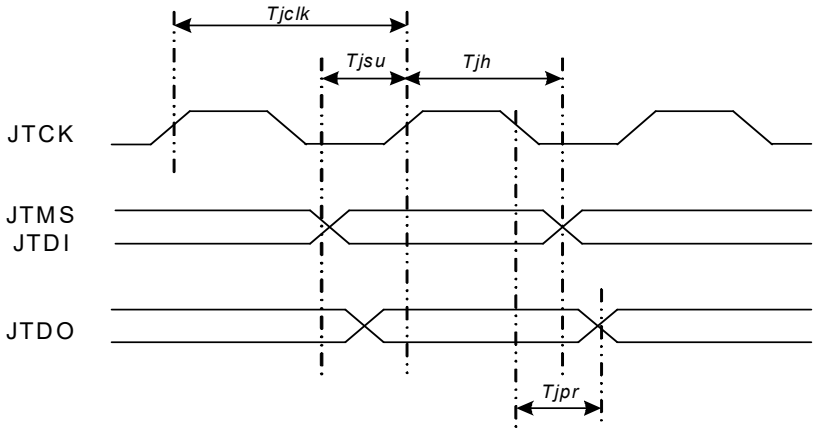


Figure 16-5. JTAG AC Timing Diagram

16.5.2.6 MDIO AC Specification

The E810 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 16-27, Figure 16-6, and Figure 16-7.

Table 16-27. MDIO I/F Timing Parameters

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

Notes:
- Table 16-27 applies to MDIO0, MDC0, MDIO1 and MDC1.
- Timing measured relative to MDC reference voltage of 2.0 V (Vih).

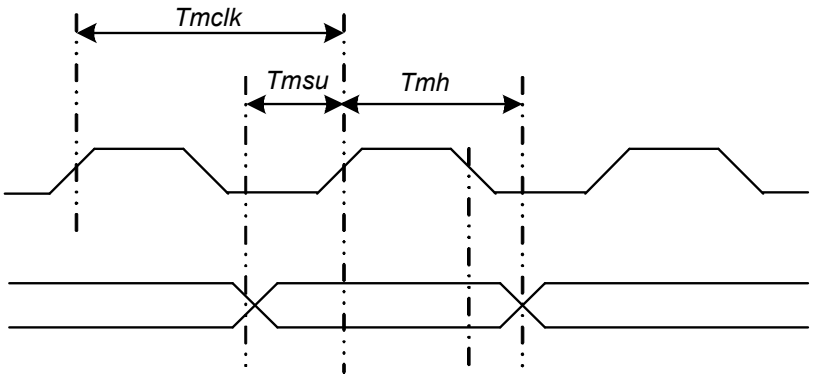


Figure 16-6. MDIO Input AC Timing Diagram

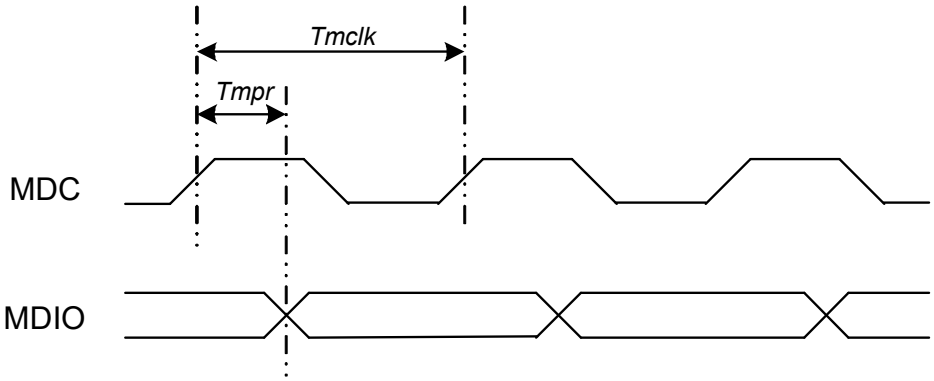


Figure 16-7. MDIO Output AC Timing Diagram

16.5.2.7 Reset Signals

For power-on indication, the E810 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. When asserting the POR_BYPASS pin, the E810 uses the LAN_PWR_GOOD pin as power-on indication. Otherwise, the E810 uses a logical OR of the internal power on detection circuit and the LAN_PWR_GOOD pin to generate the internal power-on reset signal.

Note: The timing between the power up sequence and the different reset signals is described in Section 16.3.1.1.

16.5.3 PCIe Interface AC/DC Specification

The E810 PCIe interface supports the PCIe Gen 4.0 electrical specification defined in:

- *PCIe Express Base Specification*, Revision 4.0
- *PCI Express Card Electromechanical Specification*, Revision 4.0

16.5.4 Network Interface AC/DC Specification

The E810 Ethernet interface supports a variety of operating modes, as defined in Section 3.2.2.

16.5.5 Reference Clock Specification

The E810 requires a 156.25 MHz differential reference clock input. The REFCLK input buffer supports HCSL voltage levels and contains internal AC coupling capacitors and 50 Ω termination. Best performance can be achieved by using an HCSL drive oscillator with direct connections to the REFCLK inputs (no other component connected on the circuit board).

Figure 16-8 shows a typical input waveform expected from a 1.8 V HCSL driver.

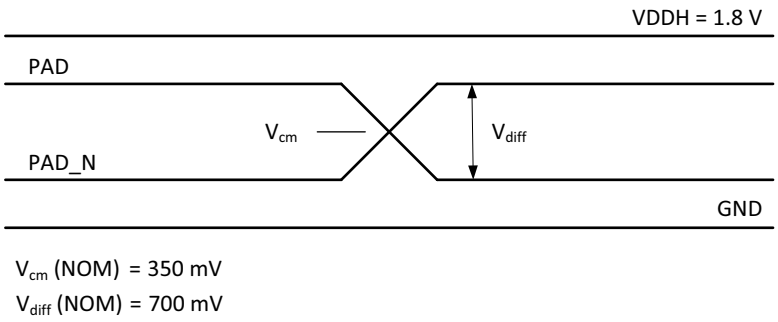


Figure 16-8. Typical HCSL Waveform

The input HCSL signal needs to stay between GND (0 V) and VDDH (1.8 V) range at all times. If the HCSL oscillator logic low V_{OL} = 0 V, the common mode voltage (V_{cm}) measured with respect to GND is half of the differential swing V_{diff}:

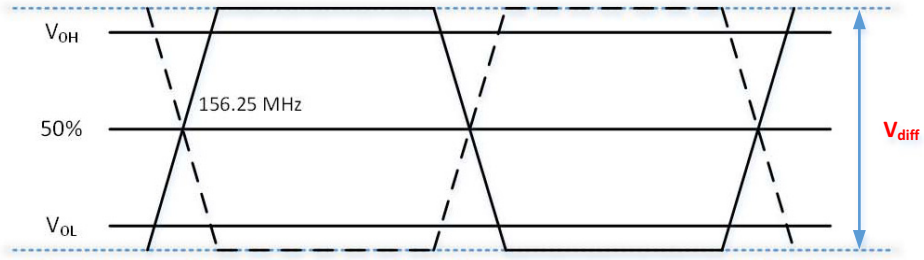
$$V_{cm} = 0.5 * V_{diff}$$

This allows for the HCSL oscillator V_{OL} to be higher than 0 V if needed, but then the max differential swing V_{diff} will need to be reduced:

- If the differential swing V_{diff} = max allowed = 1.2 V, the HCSL oscillator V_{OL} needs to = 0 V.
- If the differential swing V_{diff} < 1.2 V, the HCSL oscillator V_{OL} is allowed to be > 0 V

For example:

If V_{diff} = 800 mV and V_{cm} = 400 mV, the HCSL oscillator V_{OL} max allowed = 200 mV and HCSL oscillator V_{OH} max allowed = 1.0 V.



V_{diff} = differential swing V_{diff} = measured as a single-ended amplitude

Figure 16-9. Differential Swing V_{diff}

Table 16-28. Input Reference Clock Electrical Characteristics

Symbol	Parameter	Min	Typical	Max	Unit	Comments
f	Frequency		156.25		MHz	
Δf	Frequency Variation	-50		+50	ppm	
DC	Duty Cycle	45		55	%	
Tr	Rise Time (10% - 90%)	40	120	1000	ps	
Tf	Fall Time (90% - 10%)	40	120	1000	ps	
Vdiff	Differential Peak-to-Peak Amplitude	400		1200	mV	
Vcm	Common mode input voltage	150		600	mV	
	Start-Up time			10	ms	
	Phase Jitter		0.2	0.3	ps rms	12 KHz - 20 MHz
	Phase Noise			-110	dBc/Hz	10 KHz
				-130	dBc/Hz	100 KHz
				-150	dBc/Hz	1 MHz
				-150	dBc/Hz	10 MHz
				-150	dBc/Hz	20 MHz
				-150	dBc/Hz	100 MHz

16.6 Package Characteristics

16.6.1 Mechanical Configuration

The E810 is assembled in 25x25 mm and 21x21 mm FCGBA package with 10-layer substrates. Refer to Section 16.7 for the package mechanical drawings.

Table 16-29. Package Specifications

SKU	Body Size	Ball Count	Ball Pitch	Ball Matrix	Substrate
E810-CAM2/CAM1	25 x 25 mm	668	1 mm	48 x 28	4-2-4
E810-XXVAM2	21 x 21 mm	456	1 mm	40 x 23	4-2-4

16.6.2 Heat Sink Mechanical Load Limits

This section presents guidelines for the maximum loads associated with heat sink to PCB attachment.

For the E810-CAM2/CAM1 (25x25 mm lidded package), 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.

For the E810-XXVAM2 (21x21 mm bare-die package), the heat sink or actuator contact surface is parallel to the die backside surface. Heat sink or actuator touchdown on the die edge or corner is prohibited. Short-term loads act on both the die backside and the exposed substrate. Long-term (heat sink) loads may act solely on the die. A compliant material is used between the heat sink or actuator and the die.

Short-term and long-term loads are applied slowly to prevent dynamic loads from being created. The maximum permissible BGA compression is 15% of the assembled BGA height.

Table 16-30. Mechanical Load Limits

Maximum Load	E810-CAM2/CAM1	E810-XXVAM2
Short-term compressive load (applied for up to two minutes)	30 gm per solder ball. 65 psi on the raised portion of the lid.	30 gm per solder ball. 40 psi on the die and substrate.
Long-term compressive load	15 gm per solder ball. 30 psi on the raised portion of the lid.	15 gm per solder ball.
Dynamic load	30g shock per MIL-STD-810F, Method 516.5 Procedure II.	30g shock per MIL-STD-810F, Method 516.5 Procedure II

16.6.3 Thermal

For complete E810 package thermals, refer to the Flotherm models and Chapter 18, "Thermal Design Considerations".

Table 16-31. Typical Thermal Resistance Parameters

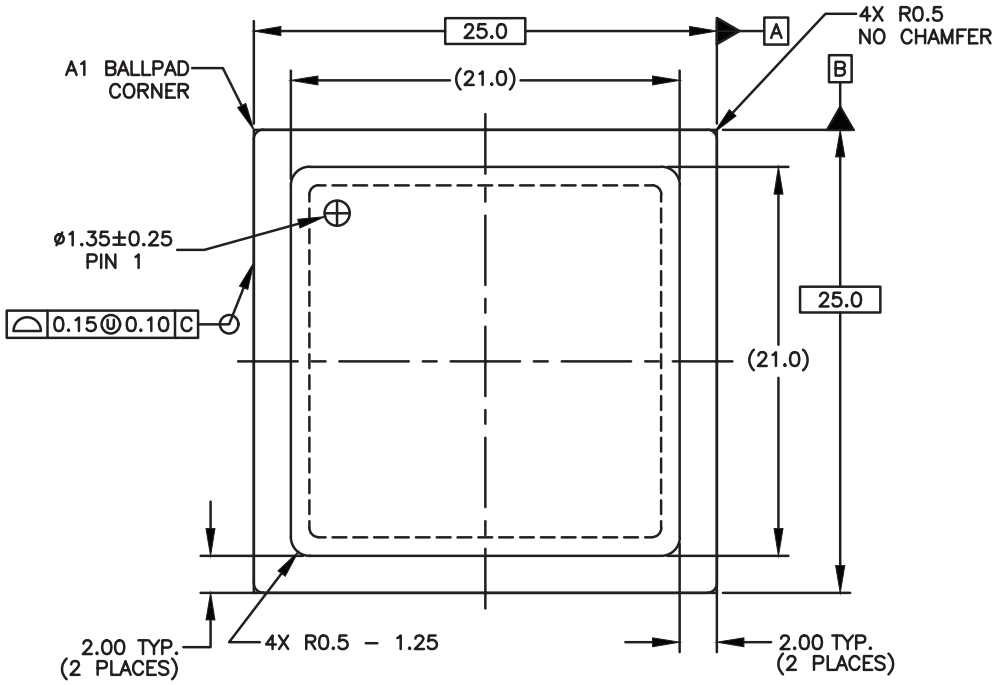
Parameter	E810-CAM2/CAM1	E810-XXVAM2
Thermal resistance, junction to case, Rjc	0.69 C/W	0.15 C/W
Thermal resistance, junction to PCB, Rjb	4.3 C/W	7.4 C/W

16.6.4 Electrical

Package electrical models are part of the IBIS files.

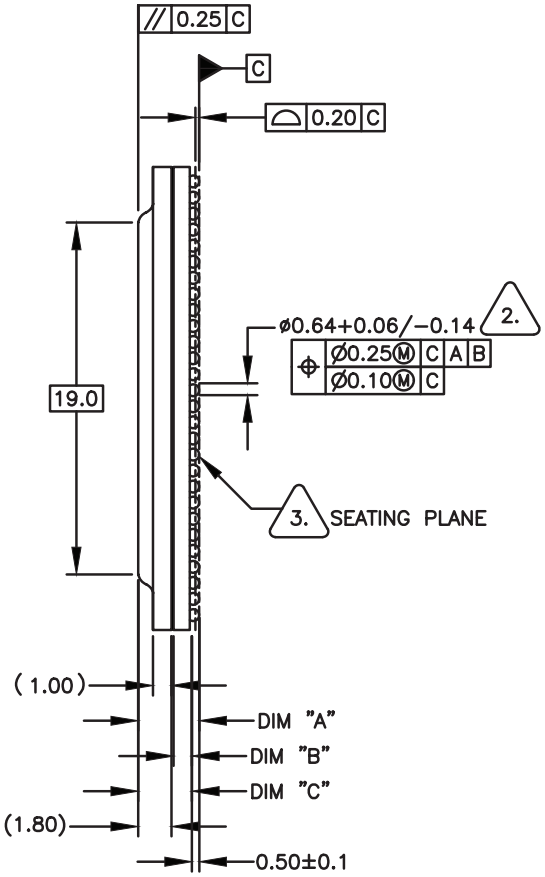
16.7 Package Mechanical Drawings

16.7.1 E810-CAM2/CAM1



4–2–4	3.51 MAX	3.05 MIN	0.876±0.088	2.78±0.131
NO. LAYERS	DIM "A"	DIM "A"	DIM "B"	DIM "C"

Figure 16-10. E810-CAM2/CAM1 Mechanical Package Diagram (Top View)



NOTES:

2. DIMENSION IS MEASURED AT THE MAXIMUM SOLDER BALL DIAMETER, PARALLEL TO PRIMARY DATUM C

3. PRIMARY DATUM C AND SEATING PLANE ARE DEFINED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS

Figure 16-11. E810-CAM2/CAM1 Mechanical Package Diagram (Side View)

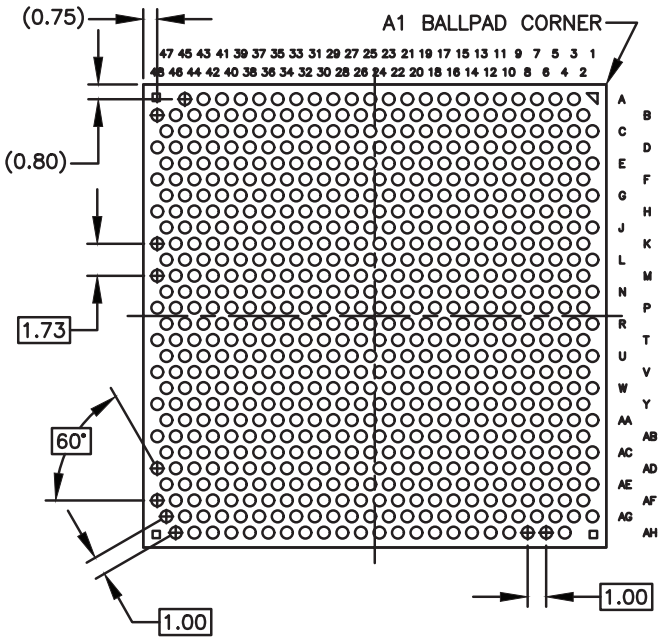
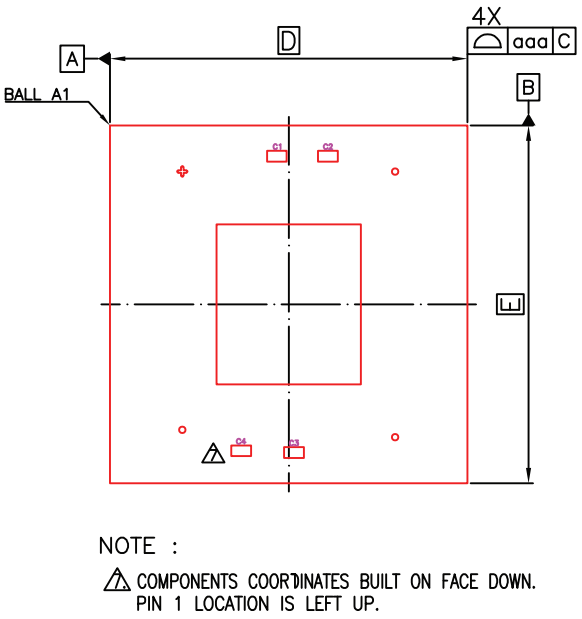


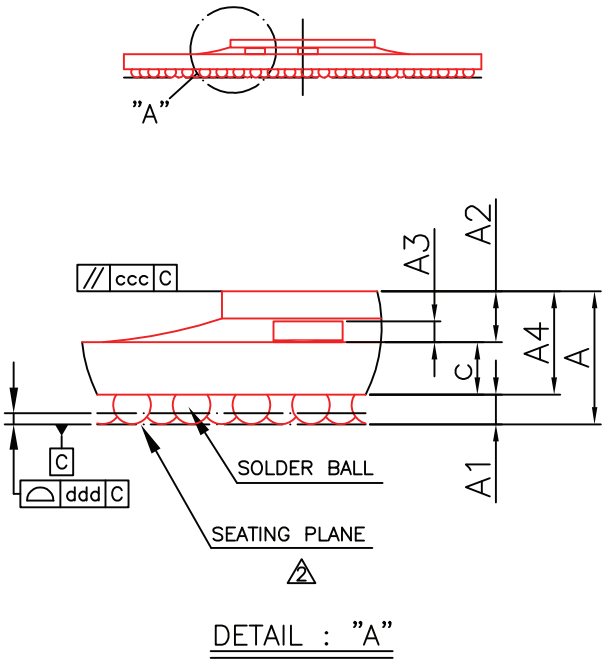
Figure 16-12. E810-CAM2/CAM1 Mechanical Package Diagram (Bottom View)

16.7.2 E810-XXVAM2



Symbol	Dimension in mm		
	MIN	NOM	MAX
A	2.040	2.231	2.420
A1	0.400	0.500	0.600
A2	0.830	0.855	0.880
A3	---	---	0.350
A4	1.639	1.731	1.823
c	0.786	0.876	0.966
D	20.90	21.00	21.10
E	20.90	21.00	21.10
D1	---	19.500	---
E1	---	19.067	---
e / f	---	1.000	---
g	---	0.500	---
h	---	0.867	---
b	0.500	0.640	0.700
aaa	0.20		
ccc	0.25		
ddd	0.20		
eee	0.25		
fff	0.10		

Figure 16-13. E810-XXVAM2 Mechanical Package Diagram (Top View)

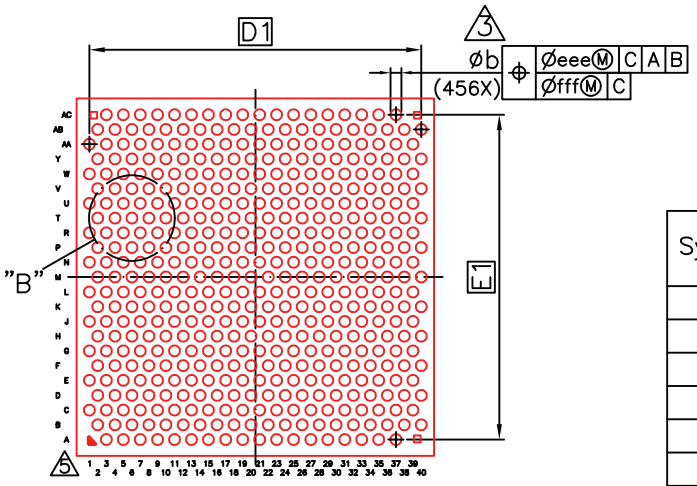


Symbol	Dimension in mm		
	MIN	NOM	MAX
A	2.040	2.231	2.420
A1	0.400	0.500	0.600
A2	0.830	0.855	0.880
A3	---	---	0.350
A4	1.639	1.731	1.823
c	0.786	0.876	0.966
D	20.90	21.00	21.10
E	20.90	21.00	21.10
D1	---	19.500	---
E1	---	19.067	---
e / f	---	1.000	---
g	---	0.500	---
h	---	0.867	---
b	0.500	0.640	0.700
aaa	0.20		
ccc	0.25		
ddd	0.20		
eee	0.25		
fff	0.10		

NOTE :

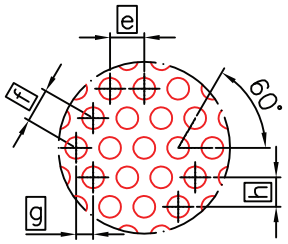
2. PRIMARY DATUM C AND SEATING PLANE ARE DEFINED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS.

Figure 16-14. E810-XXVAM2 Mechanical Package Diagram (Side View)



NOTES :

3. DIMENSION b IS MEASURED AT THE MAXIMUM SOLDER BALL DIAMETER, PARALLEL TO PRIMARY DATUM C.
5. THE PATTERN OF PIN 1 FIDUCIAL IS FOR REFERENCE ONLY.



DETAIL : "B"

Symbol	Dimension in mm		
	MIN	NOM	MAX
A	2.040	2.231	2.420
A1	0.400	0.500	0.600
A2	0.830	0.855	0.880
A3	---	---	0.350
A4	1.639	1.731	1.823
c	0.786	0.876	0.966
D	20.90	21.00	21.10
E	20.90	21.00	21.10
D1	---	19.500	---
E1	---	19.067	---
e / f	---	1.000	---
g	---	0.500	---
h	---	0.867	---
b	0.500	0.640	0.700
aaa	0.20		
ccc	0.25		
ddd	0.20		
eee	0.25		
fff	0.10		

Figure 16-15. E810-XXVAM2 Mechanical Package Diagram (Bottom View)

16.8 Devices Supported

16.8.1 Flash

The E810 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 66MHz
- 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 16-32. Supported Flash Parts

Manufacturer	128 Mbit (16 MB)
Adesto (formerly Atmel)	AT25QF128A
Winbond	W25Q128JV-xx-xx
Micron	MT25QL128ABA
Macronix	M25L12873F
GigaDevice Semiconductor	GD25B127DS

Chapter 17 Design Guidelines

17.1 Introduction

The section shows the Ethernet side interface design requirements for systems using the Intel® Ethernet Controller E810 (E810). The E810 is an integrated device that includes a 100G MAC, a x16 PCIe Gen4 block for upstream connectivity to the host processor, and eight Physical Medium Dependent (PMD) lanes capable of up to 50 Gb/s each. All speeds are dependent on the device SKU, PMD lane, and port configurations.

The E810 topologies support a variety of system configurations for native Ethernet interfaces. This section defines the supported configurations and the information required to design an E810-based system, including pin implementation and electrical specifications.

The E810 architecture allows for 2-port, 4-port, and 8-port mode configurations. These modes connect the root complex to eight PMD lanes accessible downstream. Supported E810 topology configurations are discussed in the sections that follow.

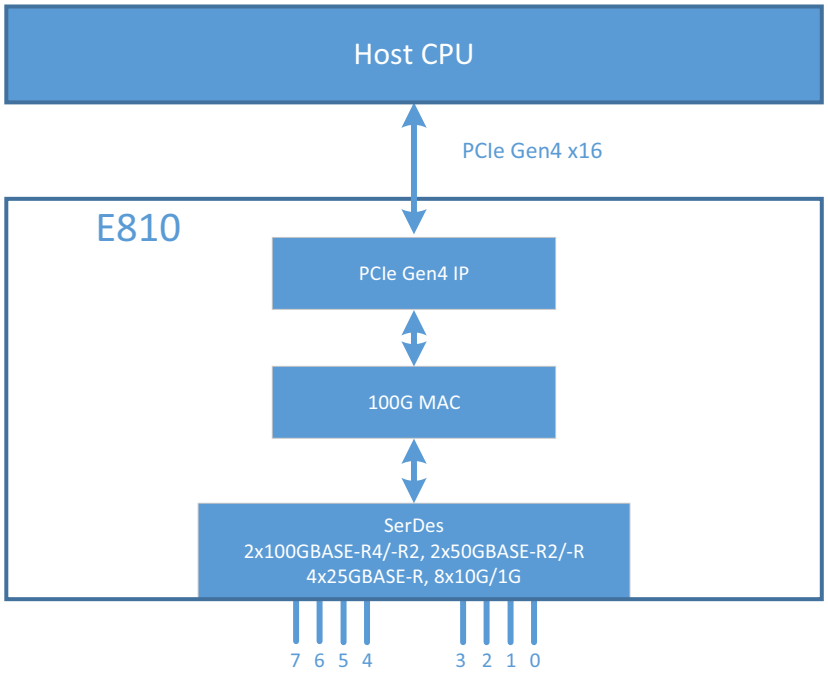


Figure 17-1. E810 Functional Block Diagram

This section shows all of the supported and validated designs for the E810 topology.

17.2 Defined Topologies

The specific product SKU used in the system determines the various configuration options, max device bandwidth, and the number of PMD lanes available. This section shows the permissible E810-CAM2 Dual 100G product configurations using eight PMD lanes with a 100G total bandwidth allocated through the device at any given time. Refer to the E810 SKU offerings for more information.

In the context of this section, groups of four PMDs make up a single "site", and each site is configured based on the chosen E810 Ethernet topology. These are referred to by the configuration names shown in Section 17.2.1.

This section defines "common" topologies. The common configurations represent those that can be implemented natively using the I/O directly from the E810 (for example, 2xSFP28 (50GBASE-R), 4xSFP28 (25GBASE-R), 1xQSFP28 (100GBASE-R4), and 2xQSFP28 (100GBASE-R4)). Within these topologies, the port rates can be individually configured to any lower supported rate.

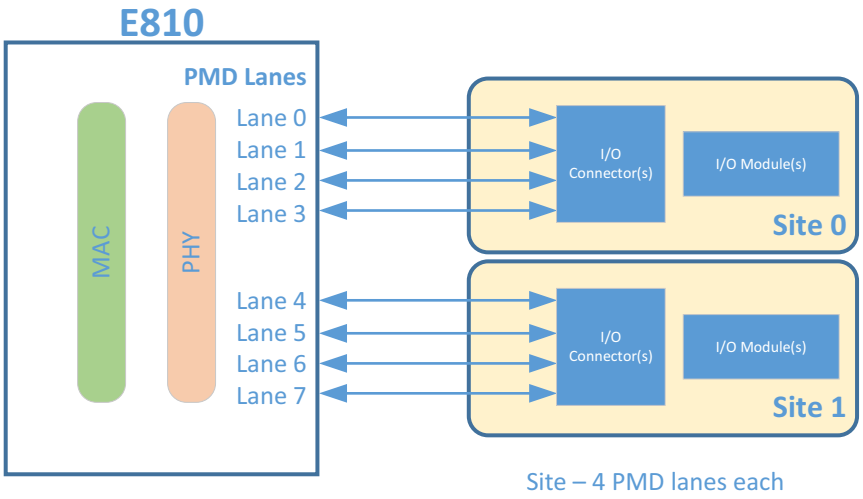


Figure 17-2. E810 Topology Components

Following is a summary of the Ethernet interfaces that the E810 provides:

- PMD lanes:
  - Each lane includes differential pairs for Tx and Rx, for a total of four pins per lane.
  - Eight PMD lanes (up to two sites).
    - Up to eight PMD lanes are available for use depending on the product SKU.
    - The maximum throughput of 100G (Topology restrictions apply. Refer to details in Section 17.2.1.)
    - The PMD lanes are denoted as [0..3] for Site 0, and [4..7] for Site 1. For a 2-port configuration with quad lane PMD modes, the Auto-Negotiation (AN) lanes exist on Lane 0, and Lane 4 by default. For single-lane configurations (e.g. 4-port and 8-port modes), the AN lanes are on a per-lane basis.

- Five MDIO/I^{2}C buses (MDIO/I^{2}C[0:4]) from the E810 that can be independently configured as MDIO or I^{2}C. For this section, the buses are assigned as shown below. The buses are shown in Section 17.2.2 for each I/O configuration.
  - MDC0_SCL0 / MDIO0_SDA0 - I^{2}C management clock and data for QSFP 0 or SFP 0.
  - MDC1_SCL1 / MDIO1_SDA1 - I^{2}C management clock and data for QSFP 1 or SFP 1.
  - MDC2_SCL2 / MDIO2_SDA2 - I^{2}C management clock and data for SFP 2.
  - MDC3_SCL3 / MDIO3_SDA3 - I^{2}C management clock and data for SFP 3.
  - MDC4_SCL4 / MDIO4_SDA4 - I^{2}C management PCA9575 I/O Expander.
- 24 Software Defined Pins (SDP):
  - 24 general purpose software definable pins are driven via the internal I/O Widget in the E810.
  - Application of these pins are defined by the link topology netlist but in general is defined as follows within this section:
    - SDP[0:7] are used as eight GPIOs.
    - SDP[8:19] are generally used as LED pins with a few crossing over for GPIO functions.
    - SDP[20:23] can be used for IEEE 1588 connections.

A summary of the I/O types that the E810 supports is presented below (see Section 17.2.3 for more information on the specific protocols supported on each I/O type):

- I/O Types:
  - SFP (Includes SFP+ and SFP28)
  - QSFP (Includes QSFP+ and QSFP28)

17.2.1 E810 Host Topology Overview

This section shows the most common host topologies that are expected to be used. Table 17-1 provides a high-level summary of the supported common configurations, and Section 17.2.2 provides the overview of each configuration.

The E810 can support up to two quads, with four PMD lanes per quad. Each quad supports multiple speed and multi-lane port configurations using 25G/lane SerDes (1x100G-R4 / 1x50G-R2 / 4x25G / 4x10G / 4x1G), or 50G/lane SerDes (1x100G-R2 / 1x50G-R). The supported common topologies are 2xSFP28 (2x50GBASE-R), 4xSFP28 (4x25GBASE-R), 1xQSFP28, and 2xQSFP28.

Implementations can use a subset of I/Os (for example, 2xSFP28 or 1xQSFP28) as a design option and is described in Section 17.3.1. A 4xSFP28 implementation also supports 2xSFP with 2x50GBASE-R lanes. PMD lanes 0 and 2 must be used for connections supporting 50GBASE-R. A 2xQSFP28 design is a superset of a 1xQSFP28 implementation with breakout capability. A design implementing 2xQSFP28 can still use the breakout capability as long as Site 1 is disabled.

Table 17-1. Supported E810 Common Configurations

			Configuration	
	PMD Lane	Speed	4x SFP	2x QSFP
Site 0	**0**	25G	4x SFP (25G) or 2x SFP (50G) on Lane 0 and 2	1x QSFP and Breakout mode (Site 1 disabled for breakout mode)
	1	25G		
	2	25G		
	3	25G		
Site 1	**4**	25G		1x QSFP
	5	25G		
	6	25G		
	7	25G		
			4-port	2-port, 4-port, 8-port

Key:	SFP LOM Design	QSFP LOM Design	Not supported in the configuration

Note: The bold lane numbers show the auto-negotiation lane for quad lane modes. The E810 supports auto-negotiation on lanes 0 and 4. System designers need to ensure these lanes are mapped to "lane 0" of the QSFP module or link partner to ensure proper connections of the AN channels. For single-lane modes, auto-negotiation occurs on all active lanes.

For each validated E810 configuration above, each PMD lane can be configured with the following speed and port width configurations when using appropriate cable combinations, as depicted in Table 17-2. The E810 supports up to 2x50G PAM4 serial lanes as well as QSFP breakout modes. For details, refer to Section 17.2.2 for configuration options.

Ports shown here with different port counts are supported by the same physical implementation from a hardware perspective, but might require a reboot and/or different NVM or netlist configuration.

Table 17-2. Supported E810 PMD Lanes and Rates for Supported Configuration Topologies

	Site #/PMD#/Supported PMD Rates									
Site	**Site 0**				**Site 1**				**Port Mode**	**Notes/Limitations**
PMD	**0***	**1**	**2**	**3**	**4***	**5**	**6**	**7**		
QSFP Lane	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**		
4x SFP	50G		50G						2	Configuration: 4xSFP (2-port and 4-port modes) ***Note:*** The 4x SFP implementation also supports 2x50GBASE-R on PMD lane 0 and 2. This configuration multiplexes the Site 1 serial lanes to Site 0.
	25G	25G	25G	25G					4	
	10G	10G	10G	10G					4	
	1G	1G	1G	1G					4	
	100M	100M	100M	100M					4	

Table 17-2. Supported E810 PMD Lanes and Rates for Supported Configuration Topologies

	Site #/PMD#/Supported PMD Rates									
Site	**Site 0**				**Site 1**				**Port Mode**	**Notes/Limitations**
PMD	**0***	**1**	**2**	**3**	**4***	**5**	**6**	**7**		
QSFP Lane	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**		
2x QSFP	100G-R4				100G-R4				2	Configuration: 2xQSFP (2-port, 4-port, and 8-port modes) ***Note:*** For the noted PMD lanes (Note 1), the configuration is only supported with topology resolution when breakout cables are plugged in. This would force the E810 to be configured in a 4-port/8-port mode configuration.
	100G-R2				100G-R2				2	
	50G-R2				50G-R2				2	
	50G				50G				2	
	25G	25G[1]			25G	25G[1]			4	
	10G	10G[1]	10G[1]	10G[1]	10G	10G[1]	10G[1]	10G[1]	8	
	1G	1G[1]	1G[1]	1G[1]	1G	1G[1]	1G[1]	1G[1]	8	
	100M	100M[1]	100M[1]	100M[1]	100M	100M[1]	100M[1]	100M[1]	8	
	50G-R2		50G-R2[1]						2	
	50G		50G[1]						2	
	25G	25G[1]	25G[1]	25G[1]					4	
	10G	10G[1]	10G[1]	10G[1]					4	
	1G	1G[1]	1G[1]	1G[1]					4	
	100M	100M[1]	100M[1]	100M[1]					4	

Key:	100G (BASE-R4/-R2)	50G (BASE-R2/-R)	25G	10G	1G/100Mb

Notes:

- All sites can be independently configured for speed and port width based on the supported mode list above via software control.
- Lane numbers denoted with an asterisk (*) show the auto-negotiation lane for quad lane modes. The E810 supports auto-negotiation on lanes 0 and 4. System designers need to ensure these lanes are mapped to "lane 0" of the QSFP module or link partner to ensure proper connections of the AN channels. For single-lane modes, auto-negotiation occurs on all active lanes.

The associated PMD lanes for each mode are called out above. Note that lanes from Quad 1 can be multiplexed over to Quad 0 when the E810 is operating in a 4-port configuration to allow a single QSFP breakout mode. Take note of the following lane speed and port capabilities:

- All 1x50G serial ports are capable of supporting 50G, 25G, 10G, 1G and 100Mb.
- All 1x25G serial ports are capable of supporting 25G, 10G, 1G and 100Mb.
- All 1x10G serial ports are capable of supporting 10G, 1G, and 100Mb.
- QSFP cable configuration support:
  - For 50G/lane configurations, 1x100GBASE-R2 and 2x50GBASE-R breakout are supported.
  - For 25G/lane configurations, 1x100GBASE-R4, 2x50GBASE-R2 and 4x25GBASE-R breakout are supported.

17.2.2 Configuration Topologies

This section presents the supported common configurations for the E810.

17.2.2.1 Configuration - 4x SFP Native

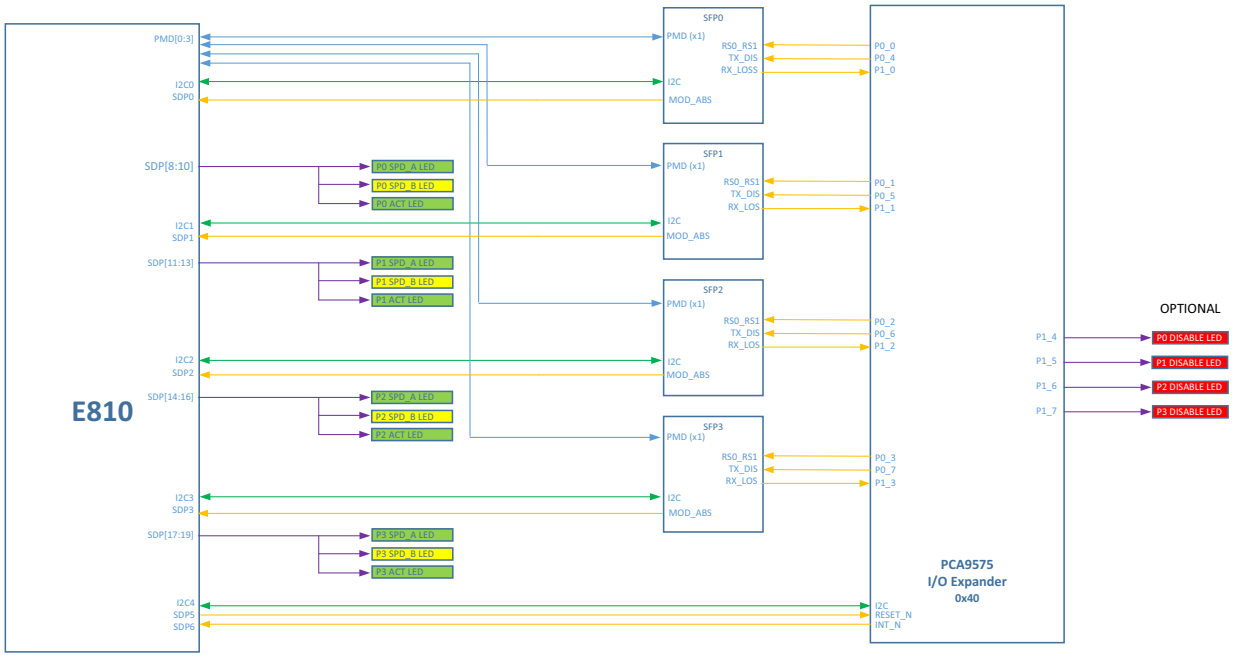


Figure 17-3. Configuration - 4x SFP Native

This configuration uses PMD lanes 0 through 3 to drive four SFP connectors on the host system. A direct I^2C connection as well as Module Absent, LOS, Rate Select, and TX Disable connections are required to each of the SFP connectors. A subset of these signals is supported on the PCA9575 I/O Expander. The link status of each port is indicated by up to three LEDs, for a total of 12 LEDs on the host system (see Section 17.5 for LED usage).

The host supports native SFI, so the E810 PMD lanes can be connected directly to the SFP connector, as shown in Figure 17-3. This figure is provided as a high-level example only, and all connection and design requirements stated within this section must be followed. All other connection requirements, such as pull-up/pull-down resistors, power/ground, LEDs, and so on are detailed in Section 17.3 through Section 17.6.

A 2x50G serial solution can also be implemented. In this case, PMD lane 0 and PMD lane 2 have the 50G serial support (SFP0 and SFP2) and must be used. Lane assignment and low-speed I/O connections for 4x25G and 2x50G modes are shown in Table 17-3.

Table 17-3. PMD Lane Assignment and Low-Speed I/O for 4x25G and 2x50G Modes

Config	Port No.	Ethernet PMD	I²C	E810 SDP/GPIO Name or Port Expander Pin Name	E810 SDP/GPIO Function or I/O Expander Function	E810 SDP/LED Name	E810 SDP/LED Function
4x25G	Port 0	L0	I2C_MDIO_CLK0/DAT0	SDP00	SFP_0_MOD_ABS	SDP08	P0_SPD_A_LED
				Port Expander I/O P1_0	SFP_0_RX_LOS	SDP09	P0_SPD_B_LED
				Port Expander I/O P0_0	SFP_0_RS0_RS1	SDP10	P0_ACT_LED
				Port Expander I/O P0_4	SFP_0_TX_DISABLE	-	-
	Port 1	L1	I2C_MDIO_CLK1/DAT1	SDP01	SFP_1_MOD_ABS	SDP11	P1_SPD_A_LED
				Port Expander I/O P1_1	SFP_1_RX_LOS	SDP12	P1_SPD_B_LED
				Port Expander I/O P0_1	SFP_1_RS0_RS1	SDP13	P1_ACT_LED
				Port Expander I/O P0_5	SFP_1_TX_DISABLE	-	-
	Port2	L2	I2C_MDIO_CLK2/DAT2	SDP02	SFP_2_MOD_ABS	SDP14	P2_SPD_A_LED
				Port Expander I/O P1_2	SFP_2_RX_LOS	SDP15	P2_SPD_B_LED
				Port Expander I/O P0_2	SFP_2_RS0_RS1	SDP16	P2_ACT_LED
				Port Expander I/O P0_6	SFP_2_TX_DISABLE	-	-
	Port3	L3	I2C_MDIO_CLK3/DAT3	SDP03	SFP_3_MOD_ABS	SDP17	P3_SPD_A_LED
				Port Expander I/O P1_3	SFP_3_RX_LOS	SDP18	P3_SPD_B_LED
				Port Expander I/O P0_3	SFP_3_RS0_RS1	SDP19	P3_ACT_LED
				Port Expander I/O P0_7	SFP_3_TX_DISABLE	-	-
2x50G	Port 0	L0	I2C_MDIO_CLK0/DAT0	SDP00	SFP_0_MOD_ABS	SDP08	P0_SPD_A_LED
				SDP02	SFP_0_RX_LOS	SDP09	P0_SPD_B_LED
				SDP05, SDP18	SFP_0_RS0_RS1	SDP10	P0_ACT_LED
				SDP14	SFP_0_TX_DISABLE	-	-
	Port 1	L2	I2C_MDIO_CLK2/DAT2	SDP01	SFP_1_MOD_ABS	SDP11	P1_SPD_A_LED
				SDP03	SFP_1_RX_LOS	SDP12	P1_SPD_B_LED
				SDP06, SDP19	SFP_1_RS0_RS1	SDP13	P1_ACT_LED
				SDP15	SFP_1_TX_DISABLE	-	-

17.2.2.2 Configuration - 2x QSFP Native

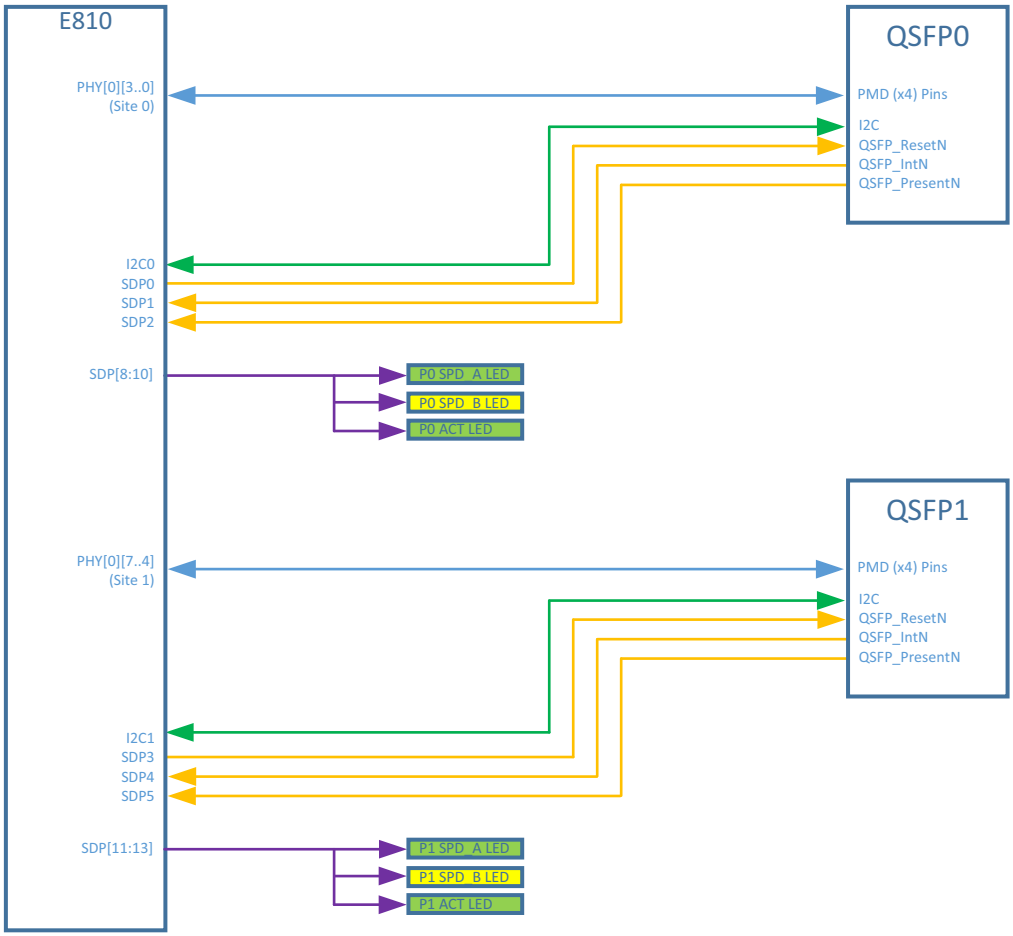


Figure 17-4. Configuration - 2x QSFP Native

This configuration uses PMD lanes 0 through 7 to drive two QSFP connections on the host system. The link status of each QSFP connection is indicated by three LEDs per port by default, for a total of six LEDs on the host system (see Section 17.5 for LED usage).

The E810 can drive a variety of QSFP protocols, so they can be connected directly to the QSFP connectors, as shown in Figure 17-4. This figure is provided as a high-level example only, and all requirements stated within this document must be followed. All connection requirements, such as pull-up/pull-down resistors, power/ground, LEDs, and so on, are detailed in Section 17.3 through Section 17.6.

This configuration supports a variety of link types, and the number of ports is dependent on the link type, since both serial and parallel link modes are available. Each PMD can operate as a single port in a serial link mode, supporting speeds of 10G and 25G, creating four ports from a single site. If the site is in a four-lane parallel link mode, all four PMDs are used to create a single port capable of 100G QSFP protocols. The device also supports the use of one 50G port per quad, with two PMDs supporting this speed. Section 17.2.3 details the variety of link modes available for QSFP connections. This configuration uses all PMD lanes available for E810 systems.

17.2.3 Supported Link Modes and Breakout Modes

The topology configurations presented in the previous section require a wide variety of native link modes from the E810. Table 17-4 provides a high-level summary of the topologies and speeds supported, including the interface to be used in each mode. Table 17-5 provides the supported QSFP breakout modes.

Table 17-4. Supported Ethernet Link Modes

Topology	Speed	Native I/F
SFP+	100M	SGMII
	1G	SGMII
	10G	SFI
SFP28	25G	25G AUI C2M
	25G	25GBASE-CR/CR1
	50G	50GBASE-CR
QSFP28	10G	SFI
	25G	25G AUI C2M
	25G	25GBASE-CR/CR1
	50G	50GBASE-CR/CR2
	50G	50G LAUI-2
	100G	CAUI-4
	100G	100GBASE-CR2/CR4

For QSFP implementations, the following breakout cases are supported:

Table 17-5. Supported QSFP Breakout Modes

Breakout	Lane Speed	Native I/F
4 x 10G	10 Gb/s	SFI
4 x 25G	25 Gb/s	25GBASE-CR/CR1
2 x 50G	25 Gb/s	50GBASE-CR2
2 x 50G	50 Gb/s	50GBASE-CR
2 x 100G	50 Gb/s	100GBASE-CR2

17.2.4 Supported Modules

Refer to Section 3.2.3, “Link Management” for details on specification compliant modules.

17.3 E810 Ethernet Signal Descriptions

The tables in this section show the Ethernet signal assignments on the E810. Annotations are provided where required. This section color codes the I/O pin groups associated with a function and are provided for a quick visual reference only. The color coding is as follows:

Color Code	Definition
SFP	High-Speed Serial (SFP+ and SFP28) related pins.
QSFP	QSFP+ and QSFP28 related pins

17.3.1 E810 High-Speed Serial

This section dictates the PMD connections between the host and the I/O.

Table 17-6. Host High-Speed Interface

	PMD[Lane]	4x SFP	2x QSFP
Site 0	PMD[**0**]	SFP0	QSFP0
	PMD[1]	SFP1	
	PMD[2]	SFP2	
	PMD[3]	SFP3	
Site 1	PMD[**4**]		QSFP1
	PMD[5]		
	PMD[6]		
	PMD[7]		

Note: The bold lane numbers show the auto-negotiation lane for quad lane modes. The E810 supports auto-negotiation on lanes 0 and 4. System designers need to ensure these lanes are mapped to "lane 0" of the QSFP module or link partner to ensure proper connections of the AN channels. For single-lane modes, auto-negotiation occurs on all active lanes.

- For all the supported implementations, the PMD connections always start with "lane 0" regardless of the number of sites implemented in the design.
- A designer might choose to implement a subset of supported sites in a given configuration. In that event, sites are removed, in order, starting with the highest numbered site.
- A designer might also choose to implement a subset of supported lanes for a given configuration.
  - For example, a 4xSFP design requiring only two SFP modules can only implement connectivity on PMD0, and PMD1.
  - The exception is when a designer is implementing two 50GBASE-R interfaces on Site 0 (See the 4xSFP column). In this case, the implemented PMD lanes are 0 and 2 per the E810 architectural definition.
- Sites can directly target an on-board I/O connector (SFP/QSFP).
- Lane numbering for each site is denoted as [0..3] for Site 0, and [4..7] for Site 1.
  - Lane and polarity swapping is dependent on the physical E810 implementation and details are not covered in this section.
  - This section assumes that both lane and polarity swapping are not implemented.

— The lane and polarity swaps are supported in the Link Topology netlist (see Section 3.3.6). Refer to Section 3.2.3, "Link Management" for details.

17.3.2 E810 Management Connections

The following table shows the I^2C and MDIO connection targets across all of the supported topologies.

Table 17-7. Host Management Connections

Pin Name	Ball #	4x SFP	2x QSFP
MDC0_SCL0 MDIO0_SDA0	AB38 AC45	SFP0	QSFP0
MDC1_SCL1 MDIO1_SDA1	Y38 AB42	SFP1	QSFP1
MDC2_SCL2 MDIO2_SDA2	AA39 AB48	SFP2	
MDC3_SCL3 MDIO3_SDA3	AB40 AA45	SFP3	
MDC4_SCL4 MDIO4_SDA4	Y40 Y48		

Note: Table 17-7 shows the superset of pins defined for the host management connections for the E810.

The I^2C SCL/SDA pins are open drain and need to be pulled up to VCC3P3 on the host system using 3.3 KΩ resistors.

A total of five configurable MDIO/I^2C buses are provided by the E810. The following MDIO and I^2C mapping are used:

- MDIO0_I2C0 bus is defined as a dedicated I^2C bus for QSFP 0 or SFP 0.
- MDIO1_I2C1 bus is defined as a dedicated I^2C bus for QSFP 1 or SFP 1.
- MDIO2_I2C2 bus is defined as a dedicated I^2C bus for SFP 2.
- MDIO3_I2C3 bus is defined as a dedicated I^2C bus for SFP 3.
- MDIO4_I2C4 bus is not used and is left as a no connect.

Example - E810 2x QSFP design:

QSFP0 is implemented on PMD lanes 0..3 (Site 0), and QSFP1 is implemented on PMD lanes 4..7 (Site 1). MDIO0_I2C0 is routed to QSFP0, and MDIO1_I2C1 is routed to QSFP1. Both interfaces operate in I^2C mode for module management.

17.3.3 E810 SDP[0:7] (GPIO) Connections

The E810 SDP [0:7] assignments share a common pin-out between all of the supported topologies and are dedicated for GPIO only. All E810 SDP [0:7] pins are defined as push-pull when used as an output.

Table 17-8. Host SDP [0:7] Connections

Pin Name	Ball #	4x SFP	2x QSFP
SDP0	V4	SFP_ModPresN_0	QSFP_ResetN_0
SDP1	T10	SFP_ModPresN_1	QSFP_IntN_0
SDP2	U3	SFP_ModPresN_2	QSFP_PresentN_0
SDP3	U7	SFP_ModPresN_3	QSFP_ResetN_1
SDP4	U11		QSFP_IntN_1
SDP5	W3	PCA9575_Reset_N	QSFP_PresentN_1
SDP6	Y4	PCA9575_Int_N	
SDP7	V10		

The pins noted in Table 17-8 have specific pull-up and pull-down requirements. These requirements are noted in Table 17-9.

Table 17-9. Host SDP[0:7] Pull-Up/Pull-Down Requirements

Signal Name	Host Requirements	Notes
SFP_ModPresN_*	10 KΩ pull up to VCC3P3	SFP ModPresN is tied to GND in the module to indicate presence.
PCA9575_Reset_N	10 KΩ pull-down to GND	Use pull-down on PCA9575 I/O Reset_N to keep I/O Expander in reset until ready to be configured.
PCA9575_Int_N	1 KΩ pull-up to VCC3P3	PCA9575 I/O Expander open drain interrupt pin requires pull-up to VCC3P3.
QSFP_ResetN_*	None	No external pull up resistor required. QSFP reset pin is internally pulled up to VCC3P3 in the module.
QSFP_IntN_*	10 KΩ pull up to VCC3P3	Interrupt pins are open drain and require a pull up.
QSFP_PresentN_*	10 KΩ pull up to VCC3P3	QSFP Present is tied to GND in the module to indicate module presence.

- SFP Module Present signals are located on SDP [0..3] for a 4x SFP implementation.
- I/O Expander Reset_N signal is located on SDP5 for a 4x SFP implementation.
- oI/O Expander Int_N signal is located on SDP6 for a 4x SFP implementation.
- QSFP Reset, Interrupt, and Present pins are assigned in that particular order and are grouped to maximize consistency across QSFP locations.
  - — QSFP0 signals use SDP[0..2].
  - — QSFP1 signals use SDP[3..5].

17.3.4 E810 SDP[8:19] (LED) Connections

The E810 has 12 additional SDPs (SDP[8:19]) for mixed function usage (LED and GPIO) between supported configurations. LED assignments share a common pin-out between the supported configurations, but be aware that the LED pins are not exclusively used for the LED function. These pins are also capable of being defined as GPIO pins. Pin configurations are noted in Table 17-10.

When the pin is used as an LED function, the connection must be connected as open drain with a pull-up on the LED, and the firmware must also be configured as open drain for LED applications. Refer to Section 17.5 for connection details. When the pin is used as a GPIO, the GPIO connection must be configured as push-pull in the firmware.

Table 17-10. Host SDP [8:19] Connections

Pin Name	Ball #	4x SFP	2x QSFP
SDP8	U9	LED0_0_spd_a	LED0_0_spd_ a
SDP9	W7	LED0_1_spd_b	LED0_1_spd_b
SDP10	W9	LED0_2_act	LED0_2_act
SDP11	V8	LED1_0_spd_a	LED1_0_spd_a
SDP12	Y8	LED1_1_spd_b	LED1_1_spd_b
SDP13	W11	LED1_2_act	LED1_2_act
SDP14	AA3	LED2_0_spd_a	
SDP15	Y10	LED2_1_spd_b	
SDP16	AA7	LED2_2_act	
SDP17	AB4	LED3_0_spd_a	
SDP18	V38	LED3_1_spd_b	
SDP19	W39	LED3_2_act	

The pins noted in Table 17-10 have specific pull-up and pull-down requirements. These requirements are noted in Table 17-11.

Table 17-11. Host SDP[8:19] Pull-Up/Pull-Down Requirements

Signal Name	Host Requirements	Notes
LED*		LEDs are connected as active low and are connected to VCC3P3 on the host through a current limiting resistor.

The pin-out description in this section assumes that the default case where three LEDs per LOM port are being used. There is the option to only use two LEDs per port, in which case the LEDn_2_act pin is not used. See Section 17.5 for more information on LED configuration and behavior options.

- SDP [8:10] are associated with QSFP 0, SFP 0.
- SDP [11:13] are associated with QSFP 1, SFP 1.
- SDP [14:16] are associated with SFP2.
- SDP [17:19] are associated with SFP3.

17.3.5 E810 SDP[20:23] (IEEE 1588) Connections

The E810 has four additional SDPs (SDP[20:23]) for IEEE 1588 timing applications. These pins can be used for custom implementations and can be routed out to a header. These pins are not used in standard SFP and QSFP implementations.

17.4 Signal Descriptions

The following sections describe the functional blocks and logical connections. Refer to the block diagrams in Section 17.2.2 for overall connection topologies. Refer to the appropriate sections for details applicable to your design.

17.4.1 High-Speed Serial

This section describes the application of the high-speed serial signals for a SFP/QSFP application.

17.4.1.1 PDM Lanes - Transmit

The transmit PMD signal names are referenced from the host side. Thus, the lane direction is directed towards the connector. This includes the following lanes:

- TX[0:3]_p / TX[0:3]_n
- TX[4:7]_p / TX[4:7]_n

For SFP/QSFP implementations, no series capacitor is required on the board as it is located within the I/O module.

17.4.1.2 PDM Lanes - Receive

The receive PMD signal names are referenced from the host side. Thus, the lane direction is directed towards the E810. This includes the following lanes:

- RX[0:3]_p / RX[0:3]_n
- RX[4:7]_p / RX[4:7]_n

For SFP/QSFP implementations, no series capacitor is required on the board as it is located within the I/O module.

17.4.1.3 SFP High-Speed Serial

For a LOM SFP design, connect the differential signals as follows:

- TX_L0 is connected to SFP0 TD.
- RX_L0 is connected to SFP0 RD.
- TX_L1 is connected to SFP1 TD.
- RX_L1 is connected to SFP1 RD.

- TX_L2 is connected to SFP2 TD.
- RX_L2 is connected to SFP2 RD.
- TX_L3 is connected to SFP3 TD.
- RX_L3 is connected to SFP3 RD.

Take note of the proper polarity connections (positive-to-positive, and negative-to-negative), as shown in Figure 17-5.

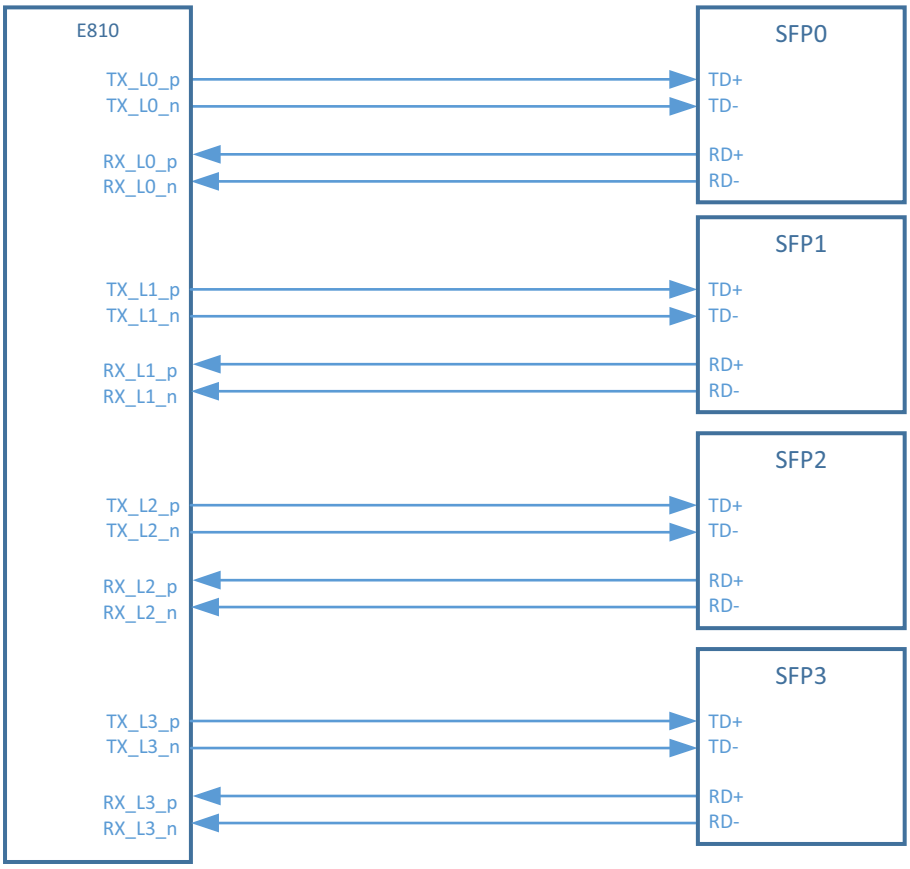


Figure 17-5. LOM SFP High Speed

17.4.1.4 QSFP High-Speed Serial

For a LOM QSFP design, connect the differential signals as follows:

- TX_L[0:3] connects to QSFP TX[1:4]
- RX_L[0:3] connects to QSFP RX[1:4]
- TX_L[4:7] connects to QSFP TX[1:4]
- RX_L[4:7] connects to QSFP RX[1:4]

Take note of the proper polarity and lane order connections, as shown in Figure 17-6.

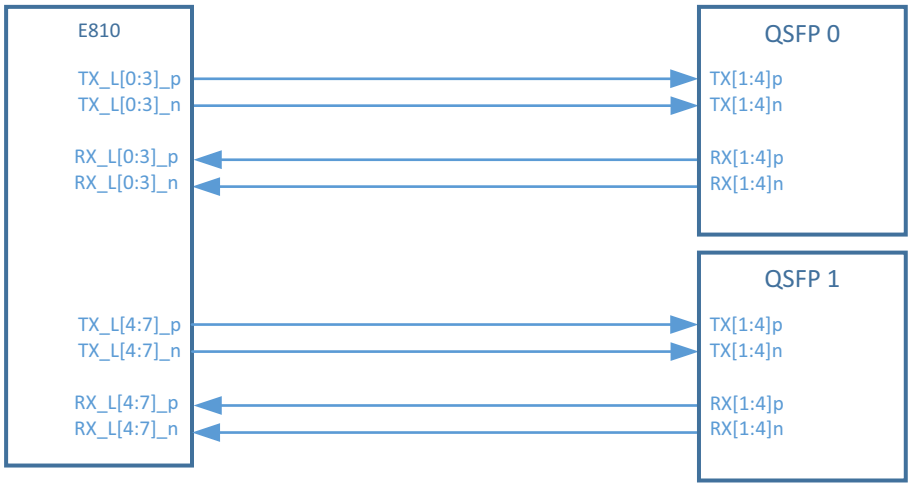


Figure 17-6. LOM QSFP High Speed

17.4.2 SFP and QSFP I/O Module Connections

This section presents the connection guidelines for configurations that implement a SFP or QSFP module(s).

17.4.2.1 SFP Cage Connections

The following connections are required for designs with SFP+/SFP28 cages. This information applies to all SFP designs. Refer to SFF-8431 for more details and specifics about the SFP form factor. Signal directions in this table are from the E810 perspective.

Table 17-12. SFP Interface

Pin Name	SFP Pin #	Type	Description
TD+/-	18, 19	A-out	Transmitter data output (towards module/link partner).
RD+/-	12, 13	A-in	Receiver data input (towards the E810).
I2C_SCL	5	Out, Pu, o/d	2-wire serial interface clock. Connect this pin directly to the E810 SCL pin appropriate for the SFP instance. Pull up to VCC3P3 on the host using a 10 KΩ resistor.
I2C_SDA	4	Inout, Pu, o/d	2-wire serial interface data. Connect this pin directly to the E810 SDA pin appropriate for the SFP instance. Pull up to VCC3P3 on the host using a 10 KΩ resistor.
Mod_ABS	6	In	Module Absent (active high). Grounded in module to indicate module presence. Connect directly between the E810 GPIO pins as defined in Section 17.3 and the SFP connector. This pin must be pulled up to VCC3P3 on the host. This pin must be connected for proper SFP operation in the E810 topology. This pin is akin to the Module Present (active low) signal in the QSFP pin-out.

Table 17-12. SFP Interface [continued]

Pin Name	SFP Pin #	Type	Description
TX Fault	2	Pu, o/d	TX Fault. When high, a laser fault of some kind is detected. Low indicates normal operation. The TX Fault pin is not used in the E810 topology and must be pulled up to VCC3P3 using a 10 KΩ resistor. If the module supports it, the TX Fault feature is available as an optional soft function to control over I^2C. Refer to SFF-8472 Address A0h, Byte 93 for details.
TX Disable	3	Out	TX disable. When high, module transmitter is disabled. When low, module transmitter is enabled. This pin is internally held high to VCC3P3 on the module. This pin must be connected for proper SFP operation in the E810 topology. If the module supports it, the TX Disable feature is also available as an optional soft function to control over I^2C. Refer to SFF-8472 Address A0h, Byte 93 for details.
Rate Select (0,1)	7, 9	Pu	Optional pin in the SFP specification. For dual rate modules, this pin defines module bandwidth control. Low indicates reduced bandwidth, while high indicates full bandwidth. This pin must be connected for proper SFP operation in the E810 topology. The pin is pulled down to ground inside the module. If the module supports it, the Rate Select feature is available as an optional soft function to control over I^2C. Refer to SFF-8472 Address A0h, Byte 93 for details.
LOS	8	Pu, o/d	Loss of signal. When high, indicates loss of signal. Normal operation when low. This pin must be connected for proper SFP operation in the E810 topology and must be pulled up to VCC3P3 using a 10 KΩ resistor. If the module supports it, the LOS the feature is available as an optional soft function to control over I^2C. Refer to SFF-8472 Address A0h, Byte 93 for details.

17.4.2.2 QSFP Cage Connections

The following connections are required for designs with QSFP+/QSFP28 cages. This information applies to all QSFP implementations. Refer to SFF-8665 for more details and specifics about the QSFP form factor. Signal directions in this table are from the E810 perspective.

Table 17-13. QSFP Interface

Pin Name	QSFP Pin #	Type	Description
TX[1..4]p/n	36, 37, 2, 3, 33, 34, 5, 6	A-out	Transmitter data output (towards module/link partner).
RX[1..4]p/n	17, 18, 21, 22, 14, 15, 24, 25	A-in	Receiver data output (towards the E810).
I2C_SCL	11	Out, o/d	2-wire serial interface clock. Pull up to VCC3P3 on the host using a 10 KΩ resistor. Connect this pin directly between the E810 and the QSFP connector.
I2C_SDA	12	Inout, o/d	2-wire serial interface data. Pull up to VCC3P3 on the host using a 10 KΩ resistor. Connect this pin directly between the E810 and the QSFP connector.
ModSelL	8	Out, Pu	Module select. Module responds to host commands when asserted low by the host. For all QSFP designs, ModSelL must be statically held low on the host. Pull down to GND on the host using a 1 KΩ resistor. QSFP modules are required to have a dedicated I^2C connection in the E810 topology. Asserting ModSelL low does not create I^2C addressing conflicts.
ResetL	9	Pu, o/d	Module reset. Module is in reset when held low by the host. ResetL is connected to the host GPIO and described in Section 17.3.3. This pin is internally held high to VCC3P3 on the module. No external pull-up resistor is required.

Table 17-13. QSFP Interface [continued]

Pin Name	QSFP Pin #	Type	Description
ModPrsL	27	Out	Module present. Grounded in module to indicate module presence. ModPrsL is connected to the host GPIO and described in Section 17.3.3. Pull up to VCC3P3 using a 10 KΩ resistor.
IntL	28	Pu	Module interrupt. IntL is connected to the host GPIO and described in Section 17.3.3. Pull up to VCC3P3 using a 10 KΩ resistor.
LPMode	31	Pu, o/d	Low power mode. Module in low power mode when pin is high. Module in high power mode when pin is low. Pin is internally pulled up to VCC3P3 in the module and defaults to low power mode. This pin is not connected on the host, and modules default to low power mode. To enable high power mode, the host must access the module EEPROM interface via I^2C writes. No external pull-up resistor is required.

17.4.3 Reset, Interrupt, and Present

A reset, interrupt, and present pin are presented towards the SFP/QSFP modules to enable device reset, interrupt detection, and presence detection. This section details the connections of the three pins.

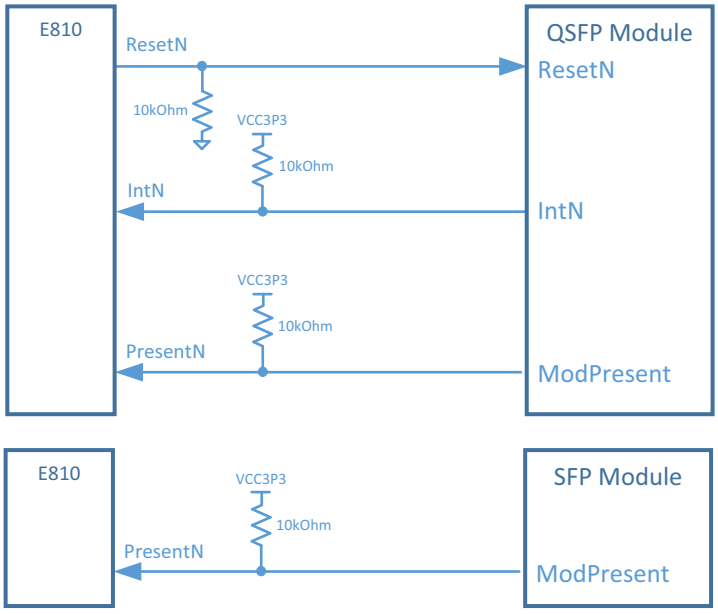


Figure 17-7. Reset, Interrupt, and Present Connection Details for SFP/QSFP Modules

17.4.3.1 ResetN

ResetN is an output pin that is driven by the host. ResetN drives the reset logic on a QSFP. The host must implement a 10 KΩ pull-down resistor on the ResetN line to ensure the device is in reset until the host is ready to initialize the downstream module or PHY.

Note: This signal is not applicable for use on SFP applications.

17.4.3.2 InterruptN

InterruptN is an input pin to the E810 that is driven low by a QSFP. This pin is asserted low when an interrupt event is detected. This pin is open drain and must be pulled up to VCC3P3 with a 10 KΩ resistor on the host.

Note: This signal is not applicable for use on SFP applications.

17.4.3.3 PresentN

PresentN is an input pin to the E810. The host must pull the pin to VCC3P3 using a 10 KΩ pull-up resistor. This pin serves as a SFP/QSFP module presence indication. When PresentN is high, no module is installed. When PresentN is low, a module is installed.

17.4.4 Management Interfaces

An I^2C bus is provided for low speed module identification and configuration for each site. The I^2C bus is a 1:1 mapping between the E810 and each module. Modules do not share an I^2C bus in this topology. The following sections describe the functionality of each of these devices and how they are connected in the E810 architecture.

Each site uses the same basic I^2C device architecture and is presented in Figure 17-8.

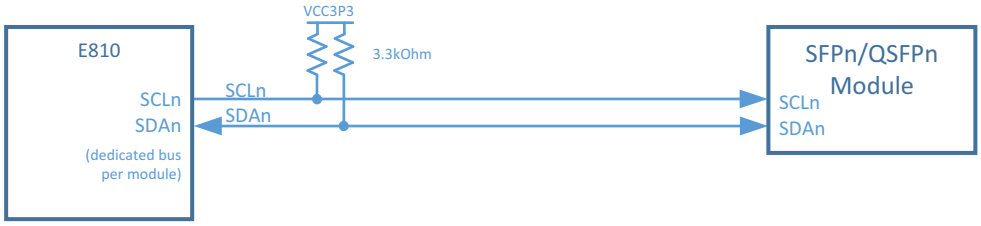


Figure 17-8. Management Interface Connection Details

The E810 Ethernet Hardware Topology defines an I^2C bus for managing SFP and QSFP modules. Per the I^2C bus specification, the SCL and SDA pins are open drain and require a pull up on the host to operate correctly. Pull up the SCL and SDA signals to VCC3P3 through a 3.3 KΩ resistor. The E810 I^2C bus architecture can operate at frequencies of 100 KHz, 400 KHz and 1 MHz at VCC3P3 voltage levels. Refer to the I^2C bus specification for the timing requirements. All SFP and QSFP modules respond to the write/read address pair of 0xA0/0xA1 (8-bit).

17.5 LED Configuration and Behavior

This section defines the default LED configuration and behavior for the various E810 architectures. There are two main design factors to take into consideration regarding LED behavior.

- Link type.
- The number of LEDs per port.

A summary of the available Link Type options is shown in Table 17-14.

Table 17-14. Link Type Options

Link Type	Lanes per Port	Supported Modes
Single PMD	1	1G/10G/25G serial link mode.
Dual PMD	2	50G parallel link mode.
Quad PMD	4	100G parallel link mode.

In general, the default LED configuration has three discrete LEDs per port. There are two speed LEDs and a single link/activity LED. This configuration is discussed in Section 17.5.1.

This section defines the LED colors and implementations that have been designed in and validated on Intel platforms. Refer to the Section 3.2.3, "Link Management" and contact your local Intel PAE representative for support and details on permissible LED configurations.

Up to 24 discrete LEDs can be implemented for any given E810 topology. The total is the sum of LEDs implemented via the E810 LED pins plus any LEDs implemented via LED drivers within a system. This constraint is based on a blink rate of 200 ms on, 200 ms off.

17.5.1 Default LED Behavior - Discrete LED Implementation

The default LED behavior is defined by using three LEDs per port. Native implementations support a three LED configuration (two speed LEDs, and one link/activity LED). A description of each pin is shown in Table 17-15. This implementation works well for the 4xSFPs and is implemented as shown in Table 17-16.

Table 17-15. Default LED Configurations - Discrete LED Implementation

<table>
<tr><th>LED Pin</th><th>LED Color</th><th>Description</th></tr>
<tr><td>Port[0:3]_Spd_A</td><td>Green</td><td>This active low bi-color LED is used as a link and link speed indicator. The expressed color indicates the port link speed. The default color coding is as follows:
<table>
<tr><th>Speed</th><th>Color</th></tr>
<tr><td>Max speed</td><td>Green</td></tr>
<tr><td>Not Max speed</td><td>Yellow</td></tr>
</table></td></tr>
</table>

Table 17-15. Default LED Configurations - Discrete LED Implementation [continued]

LED Pin	LED Color	Description
Port[0:3]_Spd_B	Yellow	The Spd_B indicator is also used for port identification through diagnostic software. When the link is up, this LEDs shall be lit and solid. This indicates that the link is established, there are no local or remote faults, and the link is ready for data packet transmission/ reception. For all single port cage implementations (SFP, QSFP), the LED indication shall operate as follows: • The LED is Green when the port is linked at its maximum speed. • The LED is Yellow when the port is linked but not operating at the highest speed. • The LED is off when no link is present. When a multi-lane cage is configured for dual port or quad port operation (e.g., QSFP breakout), the LED indication shall operate as follows: • The LED is Green when all ports of the multi-lane cage are linked at its maximum speed. • The LED is Yellow when one or more ports of the multi-lane cage are linked, but not operating at the highest speed, or one or more ports are down. • The LED is off when no link is present on all ports of the multi-lane cage.
Port[0:3]_Act	Green	Active low. The LED is illuminated when the signal is low. For all single port cage implementations (SFP, QSFP), the LED indication shall operate as follows: • The LED is off when there is no activity. • The LED should blink when there is activity. When a multi-lane cage is configured for dual port or quad port operation (e.g., QSFP breakout), the LED indication shall operate as follows: • The LED is off when there is no activity on all ports of the multi-lane cage. • The LED should blink when there is activity on one or more ports of the multi-lane cage

Table 17-16. Default LED Configurations - Discrete LED Behavior Matrix

	Speed Indication - 2 LED Pins						
	Max Speed		Not Max Speed		Link Activity - 1 LED Pin		
Port Configuration	Spd_A (Green)	Spd_B (Yellow)	Spd_A (Green)	Spd_B (Yellow)	Link (Green)	No Link (Off)	Activity (Green)
2xSFP (2x50G max)	On (50G)	Off	Off	On (<50G)	Off	Off	Blink
4xSFP28 (4x25G max)	On (25G)	Off	Off	On (<25G)	Off	Off	Blink
1xQSFP28 (1x100G max)	On (100G)	Off	Off	On (<100G)	Off	Off	Blink
1xQSFP28 (1x50G max)	On (50G)	Off	Off	On (<50G)	Off	Off	Blink
4x10GBASE-T (4x10G max)	On (10G)	Off	Off	On (<10G)	Off	Off	Blink
1x1000BASE-T (1x1G max)	On (1G)	Off	Off	On (<1G)	Off	Off	Blink

17.6 Electrical Specifications

17.6.1 Pull-Up/Pull-Down Requirements

The host and site implementations must use pull-up and pull-down resistors on I/O pins to ensure inputs and open drain I/O are in their default states. Refer to the individual signal descriptions shown in Section 17.3, and Section 17.4 for details.

For I^2C and MDIO buses, system designers need to ensure that the pull-up and pull-down values are sufficient for their implementation and that the buses do not exceed the maximum pin bus capacitance values as stated in the respective specifications.

Chapter 18 Thermal Design Considerations

18.1 Introduction

This chapter can be used as an aid when designing a thermal solution for systems implementing the Intel® Ethernet Controller E810-CAM1, the Intel® Ethernet Controller E810-CAM2, and the Intel® Ethernet Controller E810-XXVAM2. This is collectively referred to as the E810 in this chapter unless specific SKU call outs are required.

Properly designed solutions provide adequate cooling to maintain the E810 product case temperature Tcase (or junction) at or below thermal specifications. The E810 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 E810. Operation outside the functional limits can cause data corruption or permanent damage to the E810.

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.

18.2 Measuring the Thermal Conditions

This section provides a method for determining the operating temperature of the E810 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 E810. The junction temperature can be read with the Intel software tools. This document specifies a maximum allowable $T_{J\text{-}MAX}$ and $T_{C\text{-}MAX}$ values for the E810. Refer to Section 18.6 for conversion formula details.

18.3 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.

18.4 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 E810 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.

18.5 Packaging Terminology

Table 18-1. Packaging Terminology

Item	Description
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. The E810-CAM2/CAM1 package is covered with a lid to protect the on-package capacitors. The E810-XXVAM2 is an exposed die.
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).
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.
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.
Θ_{JB} (Theta JB)	Thermal resistance junction-to-board, °C/W.
Θ_{JC} (Theta JB)	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.

18.6 Thermal Specifications

The following considerations should be made when designing a proper thermal solution for the E810:

- To ensure proper operation of the E810, 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 E810 is designed to work at up to 115 °C with no reliability issues. However, functionality is not tested and the TDP may exceed the published values. Normal operation is expected to resume when the silicon returns to a max junction temperature of 105 °C.
- The E810 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 system airflow are critical to dissipate the E810'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. Good Thermal Interface Material (TIM) between the heat sink and die must be applied correctly. The E810-CAM2/CAM1 and E810-XXVAM2 FloTherm thermal models can be downloaded from rdc.intel.com.

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 E810 thermal performance.

The relationship between the parameters in Table 18-2 are as follows:

$$\Psi_{JC} = (T_J - T_C) / TDP$$

$$T_{C\text{-}MAX} = T_{J\text{-}MAX} - (\Theta_{JC} * TDP)$$

Table 18-2. E810 Thermal Specifications

Parameter	E810-CAM2	E810-CAM1	E810-XXVAM2	Notes
$T_{J\text{-}MAX}$	105 °C	105 °C	105 °C	
TDP	15.7 W	13.8 W	10.8 W	Max TDP; characterized at T_J = 105 °C. Refer to Section 16.4, "Power Dissipation" for a power down by each power rail and operational conditions.
Θ_{JC}	0.69 °C /W	0.69 °C /W	0.15 °C /W	Thermal resistance junction-to-case.
Θ_{JB}	4.3 °C /W	4.3 °C /W	7.4 °C /W	Thermal resistance junction-to-board.

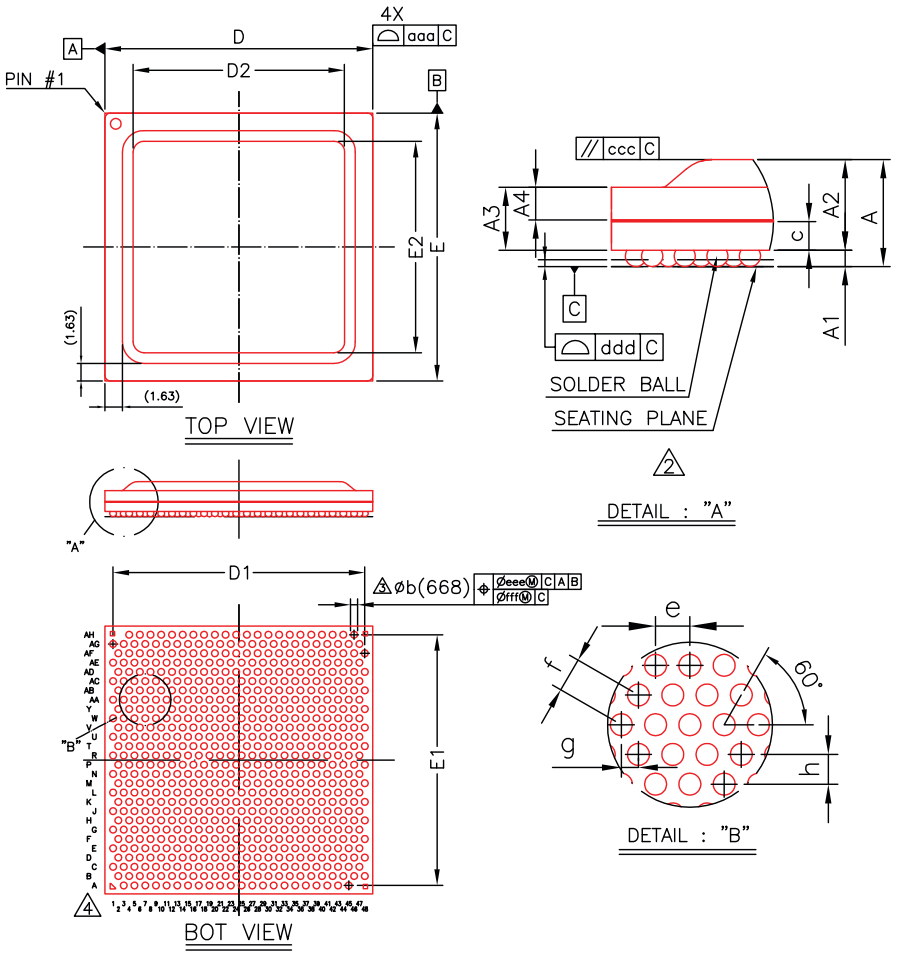
18.7 Package Mechanical Attributes

The E810 comes in two package sizes:

- **E810-CAM2/CAM1** — 25x25 mm FCBGA with integrated heat spreader (lid).
- **E810-XXVAM2** — 21x21 mm FCBGA with exposed die.

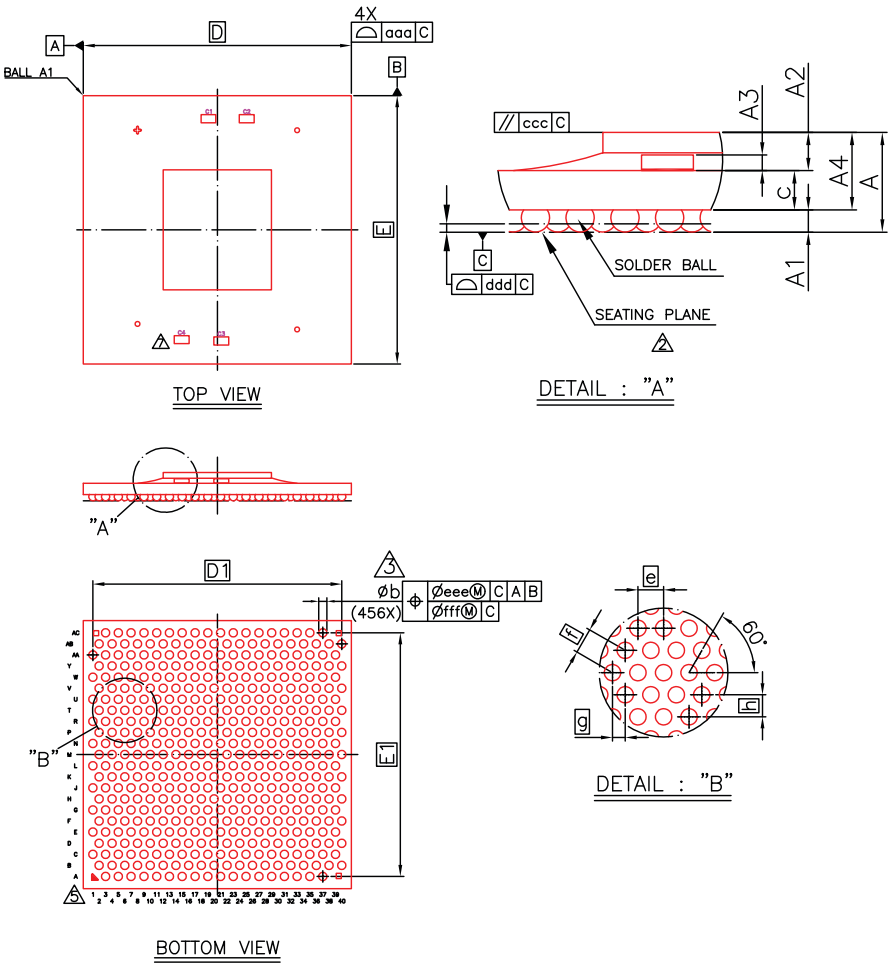
Note: Make sure official drawings are used for your detailed design.

Refer to Section 16.6.2, "Heat Sink Mechanical Load Limits" for the maximum loads for heat sink to PCB attachment.



Symbol	Dimension in mm		
	MIN	NOM	MAX
A	3.07	3.28	3.49
A1	0.40	0.50	0.60
A2	2.674	2.781	2.888
A3	---	1.931	---
A4	0.95	1.00	1.05
c	0.786	0.876	0.966
D	24.80	25.00	25.20
E	24.80	25.00	25.20
D1	---	23.50	---
E1	---	23.40	---
D2/E2	---	19.74	---
e/f	---	1.000	---
g	---	0.500	---
h	---	0.867	---
b	0.50	0.64	0.70
aaa		0.20	
ccc		0.25	
ddd		0.20	
eee		0.25	
fff		0.10	

Figure 18-1. E810-CAM2/CAM1 FCBGA Mechanical Illustration



Symbol	Dimension in mm		
	MIN	NOM	MAX
A	2.040	2.231	2.420
A1	0.400	0.500	0.600
A2	0.830	0.855	0.880
A3	---	---	0.350
A4	1.639	1.731	1.823
c	0.786	0.876	0.966
D	20.90	21.00	21.10
E	20.90	21.00	21.10
D1	---	19.500	---
E1	---	19.067	---
e / f	---	1.000	---
g	---	0.500	---
h	---	0.867	---
b	0.500	0.640	0.700
aaa	0.20		
ccc	0.25		
ddd	0.20		
eee	0.25		
fff	0.10		

NOTE :
3 COMPONENTS COORDINATES BUILT ON FACE DOWN. PIN 1 LOCATION IS LEFT UP.

DISCRETE COMPONENTS				
Item No.	X	Y	Z	COMPONENT TYPE
	COMPONENT		HEIGHT	
C1	-0.700	8.700	0.350	0204
C2	2.300	8.700	0.350	0204
C3	0.300	-8.700	0.350	0204
C4	-2.800	-8.600	0.350	0204

Figure 18-2. E810-XXVAM2 FCBGA Mechanical Illustration

Chapter 19 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 (For example, MII, MPA, Verbs, NC-SI, and so on).

Table 19-1. Definition of Terms

Term	Definition
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** — The number of bits received in error, divided by the total number of bits received.
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 E810 writes Completion events to this shared queue to make it easy for software to determine which CQ has new CQEs.
Completed, Completes, and so on	**Completion** — 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.
Connection Context	The information needed by the E810 Protocol Engine to track the state of a connection. This can include TCP- and iWARP-related information. The information includes things such as pointers to buffers, pointers to queues, pointers to CQs, sequence numbers, and so on.
Consumer	A software process that communicates with the E810. The Consumer typically consists of an application program or an OS adaptation layer that provides some OS-specific API.
CQ	**Completion Queue** — A sharable queue that 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 might be from different QPs and of differing queue types (SQs or RQs).
CQE	**Completion Queue Entry** — Info that the E810 writes onto a Completion Queue during the process of performing a Completion.
CRC	**Cyclic Redundancy Check** — An error detecting code commonly used to protect network transmissions.
CSR	**Control/Status Register** — I/O adapter registers typically accessed by the host using memory-mapped I/O requests.
DSI	**Dataplane Switch-mode Interface** — A software API for dataplane applications to allow direct attach of multiple users to device resources.
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 E810 embedded processor that handles device initialization and also device configuration based on commands received from an Admin Queue, a management interface (like NC-SI), or from a network port.
FLR	**Function-Level Reset** — A capability defined in the *PCI Express Base Specification* that enables software to quiesce and reset a particular PCI Function in a MFD, without affecting the other PCI functions therein.
FPDU	**Framed Protocol Data Unit** — The unit of data created by an MPA sender.

Table 19-1. Definition of Terms [continued]

Term	Definition
FPM	**Function Private Memory** — The E810 uses host memory as backing store for a number of context objects, such as queue state and iWARP objects. These objects are cached in the E810 Host Memory Cache (HMC). Each E810 PF or VF driver allocates host pages for this backing store according to its needs. Host pages are mapped into a virtually contiguous Private Memory address space that the HMC uses for object access. A contiguous portion of Private Memory address space is allocated for each PCI Function, and is referred to as that PCI Function's Function Private Memory (FPM). All of the host pages allocated by a given PF or VF driver are mapped into its FPM.
Frame	A unit composed of headers, data, and footers that are sent or received by a device. Also known as a *Packet*.
FSI	**Financial Services Industry** — The Financial Services Industry includes a broad range of organizations that deal with the management of money, including banks, credit card companies, insurance companies, consumer finance companies, stock brokerages, investment funds, and so on. FSI organizations often require ultra-low-latency access to financial market data such as stock feeds. Market data is typically streamed to, and distributed within these organizations using a publish/subscribe messaging model built on UDP/IP multicast transport.
HPC	**High Performance Computing** — A common term for the compute market of high performance applications.
IPG	**Inter Packet Gap** — A set of idle signals sent over the network to separate packets
IRD	**Inbound RDMA Read Queue Depth** — The maximum number of incoming outstanding RDMA Read Request Messages an E810 QP can handle. The E810 allows IRD to vary on a per QP basis.
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. • For RDMA, the LLP is DDP. • and so on.
LOM	**LAN on Motherboard** — When the E810 is integrated on a server motherboard, it qualifies as a LOM NIC.
LS	**Least Significant** — Lowest order (for example: LS bit = Least significant bit)
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 *Single-Function per Port (SFP)*.
MR	**Memory Region** — An area of memory that the RDMA Consumer wants the E810 to be able to (locally or locally and remotely) access directly in a logically contiguous fashion. A Memory Region is identified by an STag, a Base TO, and a length. A Memory Region is associated with a Physical Buffer List through the STag.
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.
Nearest Bridge Address	A multicast MAC Address used for DCBX exchange.

Table 19-1. Definition of Terms [continued]

Term	Definition
NIC	**Network Interface Controller** — An I/O adapter that enables a computer to communicate over a network.
Non-TPMR Address	**Non-Two-Port MAC Relay Address** — A multicast MAC Address used for CDCP LLDP exchange.
ORD	**Outbound RDMA Read Queue Depth** — The maximum number of outstanding RDMA Read Request Messages that the E810 can initiate from a SQ at the Data Sink. The E810 allows ORD to vary on a per QP basis.
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 are 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 might have an arbitrary length.
PBL	**Physical Buffer List** — In iWARP terminology, a Physical Buffer List can either be a Block List or a Page List. The E810 does not support Block Lists, so in the context of the E810, PBLs and Page Lists are the same thing.
PD	**Protection Domain** — A mechanism the Protocol Engine uses for tracking the association of Queue Pairs, Memory Windows, and Memory Regions. PDs are intended to be set by a Privileged Consumer to provide protection of one process from accessing another's memory, using the E810 as a proxy.
PF	**Physical Function** — A PCI Function that supports the *PCI-SIG Single Root I/O Virtualization and Sharing Specification*.
PHY Core	The 10/25/40/50/100G PHY connected to the E810.
QP	**Queue Pair** — A pair of queues that allow a Consumer to interact with the E810. The two queues are the Send Queue (sometimes called a Transmit Queue or TQ) and the Receive Queue. Each queue stores a Work Queue Element from the time it is posted until the time it is completed. The E810 LAN Engine and Protocol Engine all use QPs to interact with their various Consumers. Work Queue Elements processed by the different engines can be quite different (for example, LAN packet vs. RDMA message).
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.
Registration, Register	**Memory Registration** — The mechanism used to enable direct (local or local and remote) access by the E810 of an RDMA Consumer's Memory Region. The memory registration operation associates a Physical Buffer List to the Steering Tag (STag) returned.
RNIC	**RDMA Network Interface Controller** — The generic term for a device that implements iWARP verbs functionality. The E810 is an RNIC.
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.
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 E810 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 consisting of one or more packets. The Scheduler schedules a "quanta" in each scheduling decision.
SDPs	**Software Definable Pins** — E810 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 *Multi-Function per Port (MFP)*.
SFP	**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 E810 to a fiber optic or copper networking cable.

Table 19-1. Definition of Terms [continued]

Term	Definition
SQ	**Send Queue** — One of the two Work Queues associated with a Queue Pair. The Send Queue contains PostSQ Work Queue Elements that have specific operation types, such as Send Type, RDMA Write, or RDMA Read Type Operations, as well as STag operations such as Bind and Invalidate.
TLV	**Type, Length, Value** — A technique used to encode messages in a data communication protocol. Each message is comprised of three sequential fields: • A Type field, fixed in size and typically ~1 byte, which identifies the specific type of message and the format of information in the Value field. • A Length field, fixed in size and typically ~1 byte, which defines the length of the Value field in octets. • A Value field, variable in size, which contains all of the data specific to this message type. A protocol relevant to the E810 that uses TLV encoding is IEEE Link Layer Discovery Protocol (LLDP).
TOE	**TCP Offload Engine** — The E810 Protocol Engine integrates a TOE, which provides complete TCP/IP transport processing for RDMA connections.
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. • For RDMA verbs, an example ULP is SDP. • and so on.
VEB	**Virtual Ethernet Bridge** — This is an IEEE EVB term. A VEB is a VLAN Bridge internal to the E810 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 E810 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 Single Root I/O Virtualization and Sharing Specification*. 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.
VMDq	**Virtual Machine Devices queues** — 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.
VMM	**Virtual Machine Monitor** — A software component that allocates/exports/isolates server resources to the Virtual Machines (or System Images). Other terms for VMM in this specification: Hypervisor, Virtualization Intermediary (VI).
VPD	**Vital Product Data** — VPD is defined in the *PCI Local Bus Specification* (this is the conventional PCI specification, 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, and so on.
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 an E810 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 reconfiguration in a vendor-neutral manner.
WoL	**Wake on LAN** — 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 WoL *Advanced Power Management Wake-Up*.
WQ	**Work Queue** — One of either a Send Queue or Receive Queue.
WQE	**Work Queue Element** — A Work Request transformed by software into E810-specific format, and enqueued on a Work Queue.
WR	**Work Request** — A request to the E810 by the Consumer to perform some operation. Gets transformed by software into a Work Queue Element.

Table 19-1. Definition of Terms [continued]

Term	Definition
WRR	**Weighted Round-Robin** — A scheduling or arbitration algorithm that 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.

NOTE: ***This page intentionally left blank.***

Appendix A Factory Parsing Program

A.1 General

The following section describes the factory parsing program embedded into the E810 NVM, in particular the factory-programmed:

- Parse graph
- Protocols and frame formats
- Packet Types

The E810 factory parsing program includes a large set of frame formats that are common in various networking applications. These formats are stored in NVM as the default device programming.

Note: The factory parsing program can be modified or replaced by reprogramming the E810 parser. The device provides reprogramming of any aspect of the parsing policy, including the parse graph, frame formats, and packet types.

A.1.1 Supported Header Length

In general, the device parses headers up to the minimum of first 504 bytes of the packet, or up to 16 detected protocol headers, including payload (hereafter the "parsing depth").

Packets with headers that extend beyond the parsing depth are handled depending on the case:

- When the "next header" is located beyond the parsing depth, the packet is reported as a "partially analyzed" packet.
- When a header is chopped at the parsing depth boundary, the packet is reported as an "abort" (0xFF).

Note: Some types of headers must have a consecutive header (for example, MPLS or VLAN). For such headers, the packet is reported as an abort if there is no trailing header. The last detected protocol will always be reported as "payload".

A.2 Parse Graph

The factory parse graph represents protocol sequences supported by the factory parsing program. Nodes on the parse graph normally correspond to protocol headers such as VLAN, MPLS, IPv4, IPv6, TCP, UDP, VXLAN, and so on. Arcs on the parse graph connect nodes. Traversal over an arc from one node to another node is either unconditional (for example, L2 MAC always after VXLAN) or conditional (for example, IPv4 after L2 MAC if ETYPE = 0x0800).

The factory parse graph supports multiple overlay and tunneling technologies, including:

- MAC-in-MAC
- VXLAN, VXLAN-GPE
- Geneve

- GRE
- NSH (Service Chaining Header)
- MPLSoGRE, MPLSoUDP

Figure A-1 illustrate the E810 factory parse graph.

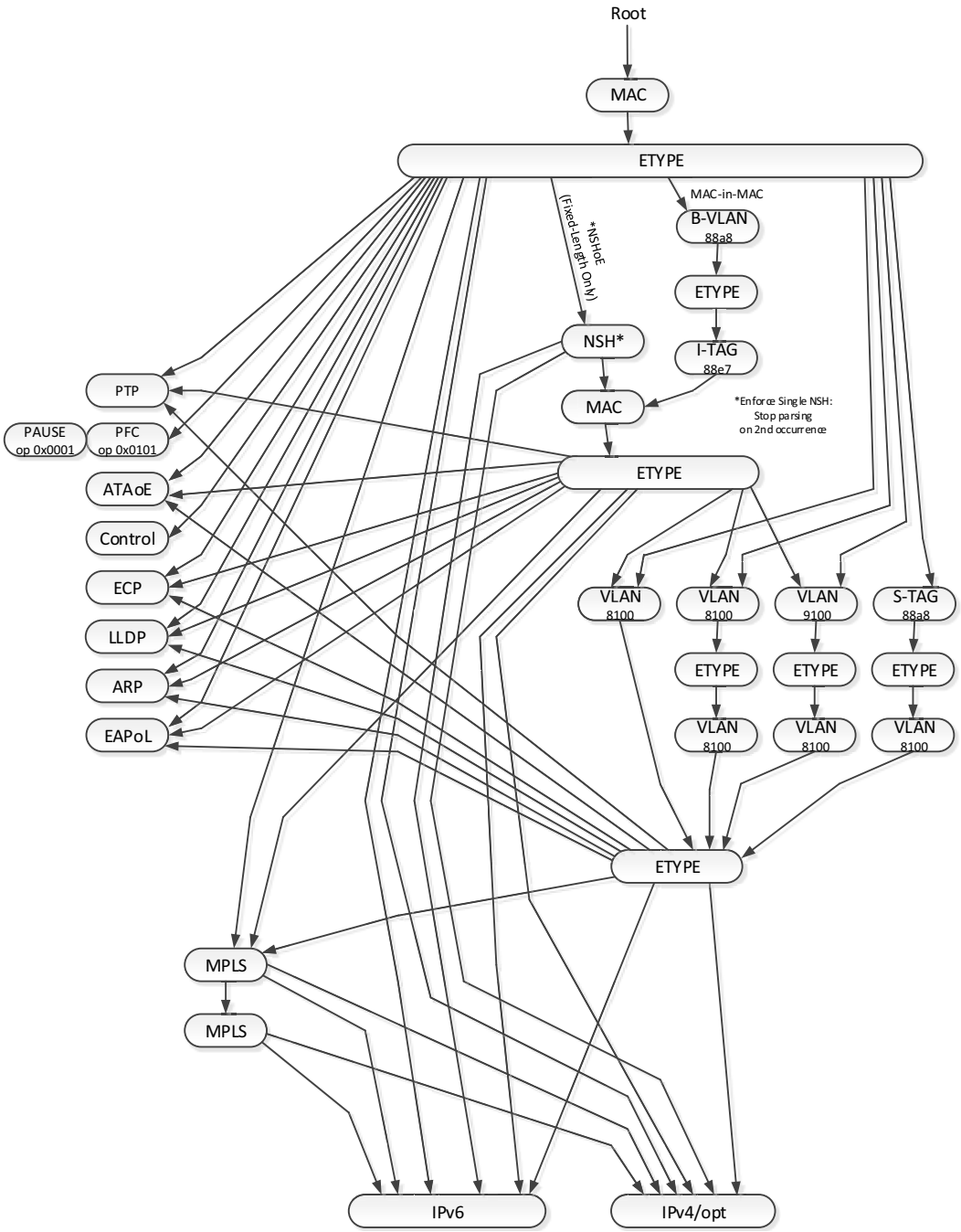


Figure A-1. Factory Parse Graph: L2+ Sections

Note: The parse graph illustrated in Figure A-1 shows the VLAN support for double VLAN mode. When supporting a single VLAN, the second occurrence of a VLAN is omitted from the parse graph.

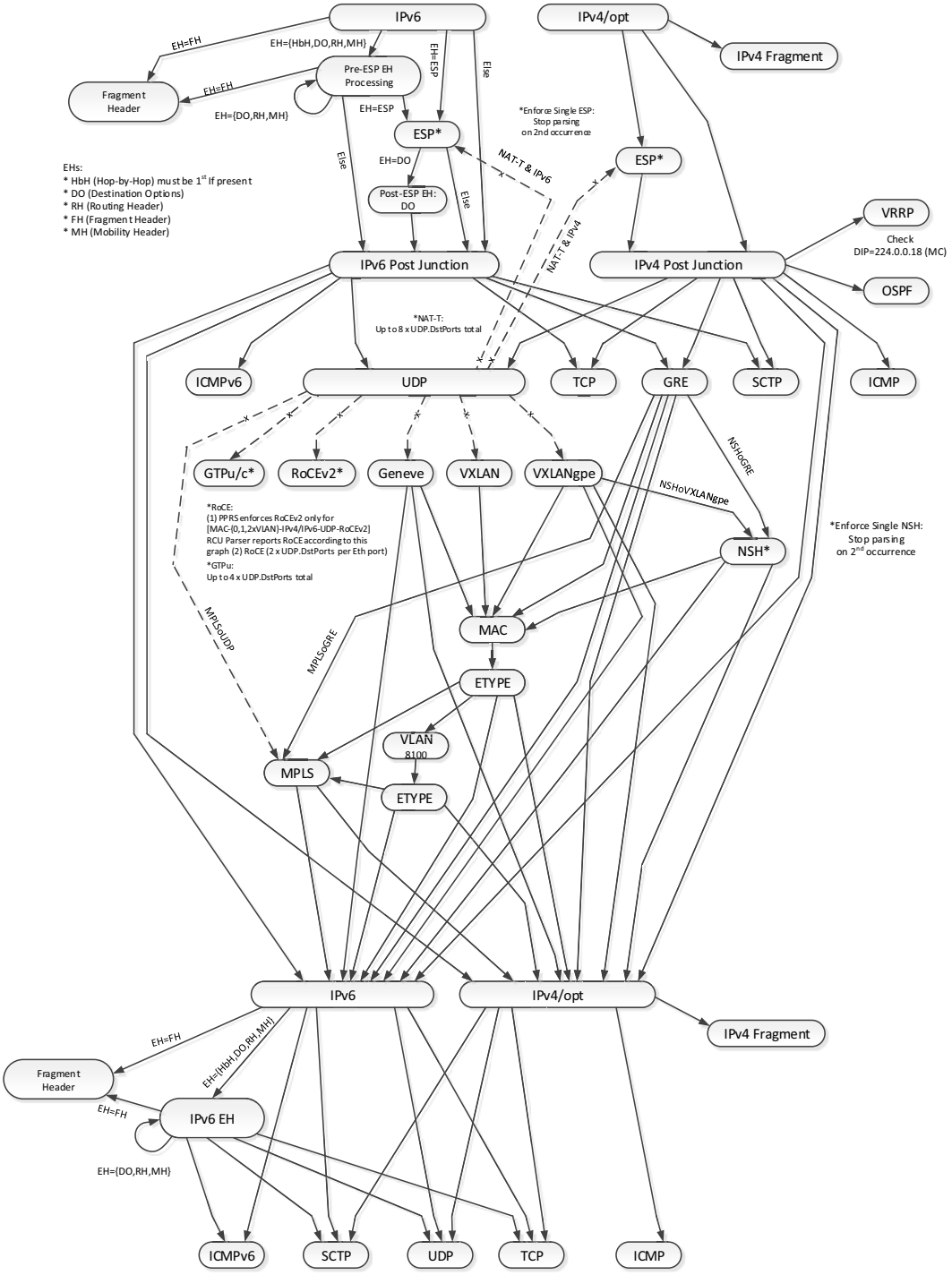


Figure A-2. Factory Parse Graph: L3+ and Inner Sections

Notes:

- The Factory Parse graph represents a superset of multiple common use cases. Some of the sequences listed on the graph are technically feasible but practically non-relevant (for example, MAC-in-MAC with STAG).
- LSO is not supported for GTPu, VRRP, OSPF packets.
- Inner VLAN insertion is not supported with inner MPLS headers.

A.3 PTYPEs

The E810 provides up to 1K programmable PTYPEs (Packet Types) for enumerating traversals over the parse graph. For example, traversal of protocol header sequences such as MAC, IPv6, IPv6, TCP, PAY4 (Payload L4). The PTYPE association of a packet is used by subsequent functions in the pipeline. For example, the ACL classifier might use the PTYPE for selecting a lookup key.

The factory-programmed PTYPEs are detailed in Table A-1.

Notes:

- Unless specifically mentioned, the packet description relates to the inner header structure.
- Table A-1 should not used to deduce which protocol IDs are generated for each PTYPE. The protocol IDs are specified in Table A-2.

Table A-1. Factory-Programmed PTYPEs

PTYPE (Dec)	PTYPE (Hex)	Packet Description	Backward Compatible to X710/XXV710/XL710
L2 Packet Types			
0	0x000	Reserved	No
1	0x001	MAC, PAY	Yes
2	0x002	Reserved	No
3	0x003	Reserved	No
4	0x004	Reserved	Yes
5	0x005	Reserved	Yes
6	0x006	MAC, LLDP	Yes
7	0x007	MAC, ECP	Yes
8	0x008	Reserved	Yes
9	0x009	Reserved	Yes
10	0x00A	MAC, EAPOL	Yes
11	0x00B	MAC, ARP	Yes
12	0x00C	Reserved	No
13	0x00D	Reserved	No
14	0x00E	Reserved	No
15	0x00F	Reserved	No
16	0x010	Reserved	No
17	0x011	Reserved	No

Table A-1. Factory-Programmed PTYPEs [continued]

PTYPE (Dec)	PTYPE (Hex)	Packet Description	Backward Compatible to X710/XXV710/XL710
18	0x012	Reserved	No
19	0x013	Reserved	No
20	0x014	Reserved	No
21	0x015	Reserved	No
Non-Tunneled IPv4			
22	0x016	MAC, IPv4FRAG, PAY	Yes
23	0x017	MAC, IPv4, PAY	Yes
24	0x018	MAC, IPv4, UDP, PAY	Yes
25	0x019	Reserved	No
26	0x01A	MAC, IPv4, TCP, PAY	Yes
27	0x01B	MAC, IPv4, SCTP, PAY	Yes
28	0x01C	MAC, IPv4, ICMP, PAY	Yes
IPv4 --> IPv4			
29	0x01D	MAC, IPv4, IPv4FRAG, PAY	Yes
30	0x01E	MAC, IPv4, IPv4, PAY	Yes
31	0x01F	MAC, IPv4, IPv4, UDP, PAY	Yes
32	0x020	Reserved	No
33	0x021	MAC, IPv4, IPv4, TCP, PAY	Yes
34	0x022	MAC, IPv4, IPv4, SCTP, PAY	Yes
35	0x023	MAC, IPv4, IPv4, ICMP, PAY	Yes
IPv4 --> IPv6			
36	0x024	MAC, IPv4, IPv6+IPv6FRAG, PAY	Yes
37	0x025	MAC, IPv4, IPv6, PAY	Yes
38	0x026	MAC, IPv4, IPv6, UDP, PAY	Yes
39	0x027	Reserved	No
40	0x028	MAC, IPv4, IPv6, TCP, PAY	Yes
41	0x029	MAC, IPv4, IPv6, SCTP, PAY	Yes
42	0x02A	MAC, IPv4, IPv6, ICMP, PAY	Yes
IPv4 --> GRE/GENEVE/VXLAN			
43	0x02B	MAC, IPv4, GRENAT, PAY	Yes
IPv4 --> GRE/GENEVE/VXLAN --> IPv4			
44	0x02C	MAC, IPv4, GRENAT, IPv4FRAG, PAY	Yes
45	0x02D	MAC, IPv4, GRENAT, IPv4, PAY	Yes
46	0x02E	MAC, IPv4, GRNAT, IPv4, UDP, PAY	Yes
47	0x02F	Reserved	No
48	0x030	MAC, IPv4, GRENAT, IPv4, TCP, PAY	Yes
49	0x031	MAC, IPv4, GRENAT, IPv4, SCTP, PAY	Yes

Table A-1. Factory-Programmed PTYPEs [continued]

PTYPE (Dec)	PTYPE (Hex)	Packet Description	Backward Compatible to X710/XXV710/XL710
50	0x032	MAC, IPv4, GRENAT, IPv4, ICMP, PAY	Yes
IPv4 --> GRE/GENEVE/VXLAN --> IPv6			
51	0x033	MAC, IPv4, GRENAT, IPv6+IPv6FRAG, PAY	Yes
52	0x034	MAC, IPv4, GRENAT, IPv6, PAY	Yes
53	0x035	MAC, IPv4, GRNAT, IPv6, UDP, PAY	Yes
54	0x036	Reserved	No
55	0x037	MAC, IPv4, GRENAT, IPv6, TCP, PAY	Yes
56	0x038	MAC, IPv4, GRENAT, IPv6, SCTP, PAY	Yes
57	0x039	MAC, IPv4, GRENAT, IPv6, ICMP, PAY	Yes
IPv4 --> GRE/GENEVE/VXLAN --> MAC			
58	0x03A	MAC, IPv4, GRENAT, MAC, PAY	Yes
IPv4 --> GRE/GENEVE/VXLAN --> MAC --> IPv4			
59	0x03B	MAC, IPv4, GRENAT, MAC, IPv4FRAG, PAY	Yes
60	0x03C	MAC, IPv4, GRENAT, MAC, IPv4, PAY	Yes
61	0x03D	MAC, IPv4, GRENAT, MAC, IPv4, UDP, PAY	Yes
62	0x03E	Reserved	No
63	0x03F	MAC, IPv4, GRENAT, MAC, IPv4, TCP, PAY	Yes
64	0x40	MAC, IPv4, GRENAT, MAC, IPv4, SCTP, PAY	Yes
65	0x41	MAC, IPv4, GRENAT, MAC, IPv4, ICMP, PAY	Yes
IPv4 --> GRE/GENEVE/VXLAN --> MAC --> IPv6			
66	0x42	MAC, IPv4, GRENAT, MAC, IPv6+IPv6FRAG, PAY	Yes
67	0x43	MAC, IPv4, GRENAT, MAC, IPv6, PAY	Yes
68	0x44	MAC, IPv4, GRENAT, MAC, IPv6, UDP, PAY	Yes
69	0x45	Reserved	No
70	0x46	MAC, IPv4, GRENAT, MAC, IPv6, TCP, PAY	Yes
71	0x47	MAC, IPv4, GRENAT, MAC, IPv6, SCTP, PAY	Yes
72	0x48	MAC, IPv4, GRENAT, MAC, IPv6, ICMP, PAY	Yes
IPv4 --> GRE/GENEVE/VXLAN --> MAC/VLAN			
73	0x49	MAC, IPv4, GRENAT, MACVLAN, PAY	Yes
IPv4 --> GRE/GENEVE/VXLAN --> MAC/VLAN --> IPv4			
74	0x4A	MAC, IPv4, GRENAT, MACVLAN, IPv4FRAG, PAY	Yes
75	0x4B	MAC, IPv4, GRENAT, MACVLAN, IPv4, PAY	Yes
76	0x4C	MAC, IPv4, GRENAT, MACVLAN, IPv4, UDP, PAY	Yes
77	0x4D	Reserved	No
78	0x4E	MAC, IPv4, GRENAT, MACVLAN, IPv4, TCP, PAY	Yes
79	0x4F	MAC, IPv4, GRENAT, MACVLAN, IPv4, SCTP, PAY	Yes
80	0x50	MAC, IPv4, GRENAT, MACVLAN, IPv4, ICMP, PAY	Yes

Table A-1. Factory-Programmed PTYPEs [continued]

PTYPE (Dec)	PTYPE (Hex)	Packet Description	Backward Compatible to X710/XXV710/XL710
IPv4 --> GRE/GENEVE/VXLAN --> MAC/VLAN --> IPv6			
81	0x51	MAC, IPv4, GRENAT, MACVLAN, IPv6+IPv6FRAG, PAY	Yes
82	0x52	MAC, IPv4, GRENAT, MACVLAN, IPv6, PAY	Yes
83	0x53	MAC, IPv4, GRENAT, MACVLAN, IPv6, UDP, PAY	Yes
84	0x54	Reserved	No
85	0x55	MAC, IPv4, GRENAT, MACVLAN, IPv6, TCP, PAY	Yes
86	0x56	MAC, IPv4, GRENAT, MACVLAN, IPv6, SCTP, PAY	Yes
87	0x57	MAC, IPv4, GRENAT, MACVLAN, IPv6, ICMP, PAY	Yes
Non-Tunneled IPv6			
88	0x58	MAC, IPv6+IPv6FRAG, PAY	Yes
89	0x59	MAC, IPv6, PAY	Yes
90	0x5A	MAC, IPv6, UDP, PAY	Yes
91	0x5B	Reserved	No
92	0x5C	MAC, IPv6, TCP, PAY	Yes
93	0x5D	MAC, IPv6, SCTP, PAY	Yes
94	0x5E	MAC, IPv6, ICMP, PAY	Yes
IPv6 --> IPv4			
95	0x5F	MAC, IPv6, IPv4FRAG, PAY	Yes
96	0x60	MAC, IPv6, IPv4, PAY	Yes
97	0x61	MAC, IPv6, IPv4, UDP, PAY	Yes
98	0x62	Reserved	No
99	0x63	MAC, IPv6, IPv4, TCP, PAY	Yes
100	0x64	MAC, IPv6, IPv4, SCTP, PAY	Yes
101	0x65	MAC, IPv6, IPv4, ICMP, PAY	Yes
IPv6 --> IPv6			
102	0x66	MAC, IPv6, IPv6+IPv6FRAG, PAY	Yes
103	0x67	MAC, IPv6, IPv6, PAY	Yes
104	0x68	MAC, IPv6, IPv6, UDP, PAY	Yes
105	0x69	Reserved	No
106	0x6A	MAC, IPv6, IPv6, TCP, PAY	Yes
107	0x6B	MAC, IPv6, IPv6, SCTP, PAY	Yes
108	0x6C	MAC, IPv6, IPv6, ICMP, PAY	Yes
IPv6 --> GRE/GENEVE/VXLAN			
109	0x6D	MAC, IPv6, GRENAT, PAY	Yes
IPv6 --> GRE/GENEVE/VXLAN --> IPv4			
110	0x6E	MAC, IPv6, GRENAT, IPv4FRAG, PAY	Yes
111	0x6F	MAC, IPv6, GRENAT, IPv4, PAY	Yes

Table A-1. Factory-Programmed PTYPEs [continued]

PTYPE (Dec)	PTYPE (Hex)	Packet Description	Backward Compatible to X710/XXV710/XL710
112	0x70	MAC, IPv6, GRENAT, IPv4, UDP, PAY	Yes
113	0x71	Reserved	No
114	0x72	MAC, IPv6, GRENAT, IPv4, TCP, PAY	Yes
115	0x73	MAC, IPv6, GRENAT, IPv4, SCTP, PAY	Yes
116	0x74	MAC, IPv6, GRENAT, IPv4, ICMP, PAY	Yes
IPv6 --> GRE/GENEVE/VXLAN --> IPv6			
117	0x75	MAC, IPv6, GRENAT, IPv6+IPv6FRAG, PAY	Yes
118	0x76	MAC, IPv6, GRENAT, IPv6, PAY	Yes
119	0x77	MAC, IPv6, GRENAT, IPv6, UDP, PAY	Yes
120	0x78	Reserved	No
121	0x79	MAC, IPv6, GRENAT, IPv6, TCP, PAY	Yes
122	0x7A	MAC, IPv6, GRENAT, IPv6, SCTP, PAY	Yes
123	0x7B	MAC, IPv6, GRENAT, IPv6, ICMP, PAY	Yes
IPv6 --> GRE/GENEVE/VXLAN --> MAC			
124	0x7C	MAC, IPv6, GRENAT, MAC, PAY	Yes
IPv6 --> GRE/GENEVE/VXLAN --> MAC --> IPv4			
125	0x7D	MAC, IPv6, GRENAT, MAC, IPv4FRAG, PAY	Yes
126	0x7E	MAC, IPv6, GRENAT, MAC, IPv4, PAY	Yes
127	0x7F	MAC, IPv6, GRENAT, MAC, IPv4, UDP, PAY	Yes
128	0x80	Reserved	No
129	0x81	MAC, IPv6, GRENAT, MAC, IPv4, TCP, PAY	Yes
130	0x82	MAC, IPv6, GRENAT, MAC, IPv4, SCTP, PAY	Yes
131	0x83	MAC, IPv6, GRENAT, MAC, IPv4, ICMP, PAY	Yes
IPv6 --> GRE/GENEVE/VXLAN --> MAC --> IPv6			
132	0x84	MAC, IPv6, GRENAT, MAC, IPv6+IPv6FRAG, PAY	Yes
133	0x85	MAC, IPv6, GRENAT, MAC, IPv6, PAY	Yes
134	0x86	MAC, IPv6, GRENAT, MAC, IPv6, UDP, PAY	Yes
135	0x87	Reserved	No
136	0x88	MAC, IPv6, GRENAT, MAC, IPv6, TCP, PAY	Yes
137	0x89	MAC, IPv6, GRENAT, MAC, IPv6, SCTP, PAY	Yes
138	0x8A	MAC, IPv6, GRENAT, MAC, IPv6, ICMP, PAY	Yes
IPv6 --> GRE/GENEVE/VXLAN --> MAC/VLAN			
139	0x8B	MAC, IPv6, GRENAT, MAC/VLAN, PAY	Yes
IPv6 --> GRE/GENEVE/VXLAN --> MAC/VLAN --> IPv4			
140	0x8C	MAC, IPv6, GRENAT, MACVLAN, IPv4FRAG, PAY	Yes
141	0x8D	MAC, IPv6, GRENAT, MACVLAN, IPv4, PAY	Yes
142	0x8E	MAC, IPv6, GRENAT, MACVLAN, IPv4, UDP, PAY	Yes

Table A-1. Factory-Programmed PTYPEs [continued]

PTYPE (Dec)	PTYPE (Hex)	Packet Description	Backward Compatible to X710/XXV710/XL710
143	0x8F	Reserved	No
144	0x90	MAC, IPv6, GRENAT, MACVLAN, IPv4, TCP, PAY	Yes
145	0x91	MAC, IPv6, GRENAT, MACVLAN, IPv4, SCTP, PAY	Yes
146	0x92	MAC, IPv6, GRENAT, MACVLAN, IPv4, ICMP, PAY	Yes
IPv6 --> GRE/GENEVE/VXLAN --> MAC/VLAN --> IPv6			
147	0x93	MAC, IPv6, GRENAT, MACVLAN, IPv6+IPv6FRAG, PAY	Yes
148	0x94	MAC, IPv6, GRENAT, MACVLAN, IPv6, PAY	Yes
149	0x95	MAC, IPv6, GRENAT, MACVLAN, IPv6, UDP, PAY	Yes
150	0x96	Reserved	No
151	0x97	MAC, IPv6, GRENAT, MACVLAN, IPv6, TCP, PAY	Yes
152	0x98	MAC, IPv6, GRENAT, MACVLAN, IPv6, SCTP, PAY	Yes
153	0x99	MAC, IPv6, GRENAT, MACVLAN, IPv6, ICMP, PAY	Yes
154-254	0x09A-0x0FE	Reserved	Yes
255	0x0FF	Parser Aborted	Yes
IPv4-UDP-GTP			
256	0x100	MAC, IPv4, GTP-C, with TEID	NO
257	0x101	MAC, IPv4, GTP-C, with/out TEID	NO
258	0x102	MAC, IPv4, GTP-U, Message Type = 0xFF	NO
259	0x103	MAC, IPv4, GTP-U, Message Type ≠ 0xFF	NO
260	0x104	Reserved for more IPv4-GTP	NO
261	0x105	Reserved for more IPv4-GTP	NO
262	0x106	Reserved for more IPv4-GTP	NO
263	0x107	Reserved for more IPv4-GTP	NO
IPv6-UDP-GTP			
26	0x108	MAC, IPv6, GTP-C, with TEID	NO
26	0x109	MAC, IPv6, GTP-C, with/out TEID	NO
26	0x10A	MAC, IPv6, GTP-U, Service Type = 0xFF	NO
26	0x10B	MAC, IPv6, GTP-U, Service Type ≠ 0xFF	NO
26	0x10C	Reserved for more IPv6-GTP	NO
26	0x10D	Reserved for more IPv6-GTP	NO
270	0x10E	Reserved for more IPv6-GTP	NO
271	0x10F	Reserved for more IPv6-GTP	NO
Other			
272	0x110	MAC, IPv4, VRRP	NO
273	0x111	MAC, IPv4, OSPF	NO
274	0x112	MAC, IPv6, VRRP	NO
275	0x113	MAC, IPv6, OSPF	NO

Table A-1. Factory-Programmed PTYPEs [continued]

PTYPE (Dec)	PTYPE (Hex)	Packet Description	Backward Compatible to X710/XXV710/XL710
276	0x114	MAC, ATAoE	NO
277	0x115	Reserved	NO
278	0x116	MAC, Control ETYPE 0x8808 (PAUSE, PFC, other control frames)	NO
279-1023		Reserved	No

Notes:
- "GRENAT" means GRE or any UDP tunnel other than UDP-NAT-ESP. This means that several different traversal paths on the parse graph might result in the same PTYPE, yet will each have a different composition of protocol IDs and flags.
- "MAC" is the first MAC protocol in packet.

A.4 Protocol IDs

The E810 provides up to 256 programmable Protocol IDs (Protocol Identification) for enumerating protocol headers or other points of interest in the parse graph.

The factory-programmed Protocol IDs are detailed in Table A-2.

Table A-2. Factory-Programmed Protocol IDs

PTYPE (Dec)	PTYPE (Hex)	Mnemonic	Protocol	Layer	Position	Description
0	0x000	Reserved	-----	-----	-----	Reserved.
1	0x001	MAC.o.[1]	MAC	Outer	1st	Outer or single MAC.
2	0x002	MAC.o.[2]	MAC	Outer	2nd	Outer 2nd MAC (can exist only if inner exists).
3	0x003	Reserved	-----	-----	-----	Reserved.
4	0x004	MAC.i.[L]	MAC	Inner	Last	Inner Last MAC.
5	0x005	Reserved	-----	-----	-----	Reserved.
6	0x006	Reserved	-----	-----	-----	Reserved.
7	0x007	MAC-in-MAC.o.[1]	MAC-in-MAC	Outer	1st	Outer 1st MAC-in-MAC. ***Note:*** This is reported when "STAG+ITAG" (MAC-in-MAC) found. In this case, STAG protID is not reported (even though an "STAG" is found in the pkt).
8	0x008	Reserved	-----	-----	-----	Reserved
9	0x009	ETYPE.o.[L]	ETYPE	Outer	Last	Outer Last ETYPE. ***Note:*** The offset points to the ETYPE field.
10	0x00A	ETYPE.i.[L]	ETYPE	Inner	Last	Inner Last ETYPE. ***Notes:*** The offset points to the ETYPE field
11	0x00B	Reserved	-----	-----	-----	Reserved.
12	0x00C	Reserved	-----	-----	-----	Reserved.
13	0x00D	Reserved	-----	-----	-----	Reserved.
14	0x00E	Reserved	-----	-----	-----	Reserved.

Table A-2. Factory-Programmed Protocol IDs [continued]

PTYPE (Dec)	PTYPE (Hex)	Mnemonic	Protocol	Layer	Position	Description
15	0x00F	Payload	-----	-----	-----	Payload. ***Note:*** Will be reported even if no payload in the pkt.
16	0x010	VLAN.o.[1]	VLAN	Outer	1st	Outer external VLAN. ***Note:*** This is EVLAN (0x8100 or 0x9100) or STAG (0x88a8) (flags distinguish between them)
17	0x011	VLAN.o.[2]	VLAN	Outer	2nd	Outer VLAN. ***Note:*** This is always VLAN (0x8100).
18	0x012	VLAN.i.[1]	VLAN	Inner	1st	Inner 1st VLAN.
19	0x013	Reserved	-----	-----	-----	Reserved.
20	0x014	Reserved	-----	-----	-----	Reserved.
21	0x015	Reserved	-----	-----	-----	Reserved.
22	0x016	Reserved	-----	-----	-----	Reserved.
23	0x017	Reserved	-----	-----	-----	Reserved.
24	0x018	Reserved	-----	-----	-----	Reserved.
25	0x019	Reserved	-----	-----	-----	Reserved.
26	0x01A	Reserved	-----	-----	-----	Reserved.
27	0x01B	MPLS.o.[L-1]	MPLS	Outer	Last-1	Outer Last-1 MPLS.
28	0x01C	MPLS.o.[L]	MPLS	Outer	Last	Outer Last MPLS. ***Note:*** Outer last or outer single.
29	0x01D	MPLS.i.[L]	MPLS	Inner	Last	Inner Last MPLS.
30	0x01E	Reserved	-----	-----	-----	Reserved.
31	0x01F	Reserved	-----	-----	-----	Reserved.
32	0x020	IPv4.o.[1]	IPv4	Outer	1st	Outer 1st IPv4. ***Note:*** Outer or single.
33	0x021	IPv4.i.[L]	IPv4	Inner	Last	Inner Last IPv4. • Only in tunneled packets.
34	0x022	Reserved	-----	-----	-----	Reserved.
35	0x023	Reserved	-----	-----	-----	Reserved.
36	0x024	Reserved	-----	-----	-----	Reserved.
37	0x025	Reserved	-----	-----	-----	Reserved.
38	0x026	Reserved	-----	-----	-----	Reserved.
39	0x027	Reserved	-----	-----	-----	Reserved.
40	0x028	IPv6.o.[1]	IPv6	Outer	1st	Outer 1st IPv6. ***Note:*** Outer or single.
41	0x029	IPv6.i.[L]	IPv6	Inner	Last	Inner Last IPv6. ***Note:*** Only in tunneled packets.
42	0x02A	Reserved	-----	-----	-----	Reserved.
43	0x02B	Reserved	-----	-----	-----	Reserved.
44	0x02C	Reserved	-----	-----	-----	Reserved.

Table A-2. Factory-Programmed Protocol IDs [continued]

PTYPE (Dec)	PTYPE (Hex)	Mnemonic	Protocol	Layer	Position	Description
45	0x02D	Reserved	-----	-----	-----	Reserved.
46	0x02E	Reserved	-----	-----	-----	Reserved.
47	0x02F	IPv6FRAG.o.[1]	IPv6FRAG	-----	1st	1st IPv6 Frag.
48	0x030	Reserved	-----	-----	-----	Reserved.
49	0x031	TCP.i.[L]	TCP	Inner	Last	Inner Last TCP.
50	0x032	Reserved	-----	-----	-----	Reserved.
51	0x033	Reserved	-----	-----	-----	Reserved.
52	0x034	UDP.o.[1]	UDP	Outer	1st	Outer 1st UDP. ***Note:*** Only in tunneled packets.
53	0x035	UDP.i.[L]	UDP	Inner	Last	Inner Last UDP. ***Note:*** Inner UDP in tunneled packets or single UDP in non-tunneled packets.
54	0x036	Reserved	-----	-----	-----	Reserved.
55	0x037	Reserved	-----	-----	-----	Reserved.
56	0x038	Reserved	-----	-----	-----	Reserved.
57	0x039	Reserved	-----	-----	-----	Reserved.
58	0x03Q	Reserved	-----	-----	-----	Reserved.
59	0x03B	Reserved	-----	-----	-----	Reserved.
60	0x03C	Reserved	-----	-----	-----	Reserved.
61	0x03D	Reserved	-----	-----	-----	Reserved.
62	0x03E	Reserved	-----	-----	-----	Reserved.
63	0x03F	Reserved	-----	-----	-----	Reserved.
64	0x40	GRE.o.[1]	GRE	Outer	1st	Outer 1st GRE.
65	0x41	Reserved	-----	-----	-----	Reserved.
66	0x42	Reserved	-----	-----	-----	Reserved.
67	0x43	Reserved	-----	-----	-----	Reserved.
68	0x44	Reserved	-----	-----	-----	Reserved.
69	0x45	Reserved	-----	-----	-----	Reserved.
70	0x46	Reserved	-----	-----	-----	Reserved.
71	0x47	Reserved	-----	-----	-----	Reserved.
72	0x48	Reserved	-----	-----	-----	Reserved.
73	0x49	Reserved	-----	-----	-----	Reserved.
74	0x4A	Reserved	-----	-----	-----	Reserved.
75	0x4B	Reserved	-----	-----	-----	Reserved.
76	0x4C	Reserved	-----	-----	-----	Reserved.
77	0x4D	Reserved	-----	-----	-----	Reserved.
78	0x4E	Reserved	-----	-----	-----	Reserved.
79	0x4F	Reserved	-----	-----	-----	Reserved.

Table A-2. Factory-Programmed Protocol IDs [continued]

PTYPE (Dec)	PTYPE (Hex)	Mnemonic	Protocol	Layer	Position	Description
80	0x50	Reserved	-----	-----	-----	Reserved.
81	0x51	Reserved	-----	-----	-----	Reserved.
82	0x52	Reserved	-----	-----	-----	Reserved.
83	0x53	Reserved	-----	-----	-----	Reserved.
84	0x54	NSH.[1]	NSH	N/A	1st	1st NSH.
85	0x55	Reserved	-----	-----	-----	Reserved.
86	0x56	Reserved	-----	-----	-----	Reserved.
87	0x57	Reserved	-----	-----	-----	Reserved.
88	0x58	ESP.[1]	ESP	N/A	1st	1st ESP.
89	0x59	Reserved	-----	-----	-----	Reserved.
90	0x5A	Reserved	-----	-----	-----	Reserved.
91	0x5B	Reserved	-----	-----	-----	Reserved.
92	0x5C	Reserved	-----	-----	-----	Reserved.
93	0x5D	Reserved	-----	-----	-----	Reserved.
94	0x5E	Reserved	-----	-----	-----	Reserved.
95	0x5F	Reserved	-----	-----	-----	Reserved.
96	0x60	SCTP.i.[L]	SCTP	Inner	Last	Inner Last SCTP.
97	0x61	Reserved	-----	-----	-----	Reserved.
98	0x62	ICMP.i.[L]	ICMP	Inner	Last	Inner Last ICMP.
99	0x63	Reserved	-----	-----	-----	Reserved.
100	0x64	ICMPv6.i.[L]	ICMPv6	Inner	Last	Inner Last ICMPv6.
101	0x65	VRRP	VRRP	N/A	1st	1st VRRP.
102	0x66	OSPF	OSPF	N/A	1st	1st OSPF.
103	0x67	Reserved	-----	-----	-----	Reserved.
104	0x68	Reserved	-----	-----	-----	Reserved.
105	0x69	Reserved	-----	-----	-----	Reserved.
106	0x6A	Reserved	-----	-----	-----	Reserved.
107	0x6B	Reserved	-----	-----	-----	Reserved.
108	0x6C	Reserved	-----	-----	-----	Reserved.
109	0x6D	Reserved	-----	-----	-----	Reserved.
110	0x6E	Reserved	-----	-----	-----	Reserved.
111	0x6F	Reserved	-----	-----	-----	Reserved.
112	0x70	Reserved	-----	-----	-----	Reserved.
113	0x71	Reserved	-----	-----	-----	Reserved.
114	0x72	ATAoE.o.[1]	ATAoE	Outer	1st	Outer 1st ATAoE.
115	0x73	Reserved	-----	-----	-----	Reserved.
116	0x74	CONTROL.o.[1]	Control	Outer	1st	Outer 1st control.

Table A-2. Factory-Programmed Protocol IDs [continued]

PTYPE (Dec)	PTYPE (Hex)	Mnemonic	Protocol	Layer	Position	Description
117	0x75	LLDP.o.[1]	LLDP	Outer	1st	Outer 1st LLDP.
118	0x76	ARP.o.[1]	ARP	Outer	1st	Outer 1st ARP.
119	0x77	TIMESYNC.o.[1]	TIMESYNC	Outer	1st	Outer 1st TIMESYNC.
120	0x78	EAPOL.[1]	EAPOL	Outer	1st	Outer 1st EAPOL.
121	0x79	Reserved	-----	-----	-----	Reserved.
122	0x7A	Reserved	-----	-----	-----	Reserved.
123	0x7B	Reserved	-----	-----	-----	Reserved.
124	0x7C	Reserved	-----	-----	-----	Reserved.
125	0x7D	Reserved	-----	-----	-----	Reserved.
126	0x7E	Reserved	-----	-----	-----	Reserved.
127	0x7F	Reserved	-----	-----	-----	Reserved.
128-254	0x80-0xFE	Reserved	-----	-----	-----	Reserved.
255	0XFF	Not a protocol	-----	-----	-----	signals that this is not a valid protocol.

A.5 Frame Formats

The following section details the factory programmed frame formats in the E810.

A.5.1 Layer 2

The following section details the L2 Native Frame formats.

A.5.1.1 Ethernet II (DIX)

The Ethernet II (DIX) frame format is illustrated in Figure A-3.

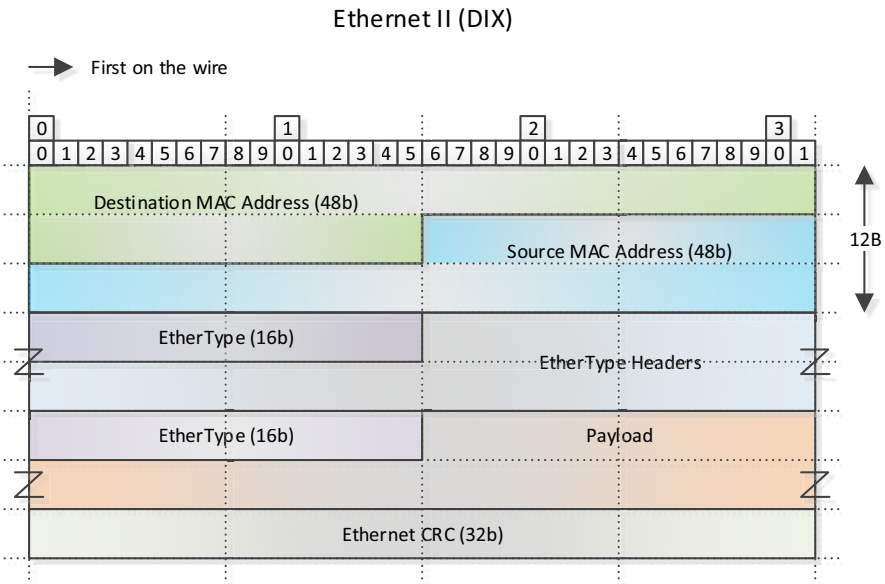


Figure A-3. Ethernet II (DIX) Frame Format

A.5.1.2 EtherTypes (L2 Tags)

Table A-3 lists the factory preset EtherType headers.

Table A-3. Factory Preset EtherTypes

EtherType	Protocol	Notes
0x88A8	B-VLAN (MAC-in-MAC)	Followed by ITag. Note that B-VLAN ETYPE is identical to STag ETYPE.
0x894F	NSHoE	NSH over Ethernet in the outer L2 section of the packet.
0x88A8	STag	IEEE 802.1Q clause 9
0x88E7	ITag	IEEE 802.1Q clause 9.7
0x8100	VLAN	IEEE 802.1Q clause 9
0x9100	VLAN	IEEE 802.1Q clause 9
0x8808	Ethernet Flow Control	PAUSE, PFC
0x8809	LACP	Link Aggregation Control Protocol
0x88F7	PTP (1588)	Precision Time Protocol (PTP) This packet type is not present in the factory parse graph.
0x88CC	LLDP	IEEE P802.1AB
0x8940	ECP	IEEE 802.1Qbg
0x0806	ARP	See Section A.5.3.8.
0x8863	PPPoE (Discovery stage)	
0x8864	PPPoE (Session stage)	
0x888E	EAPoL	IEEE 802.1X
0x88A2	ATAoE	AoE (ATA over Ethernet)

Table A-3. Factory Preset EtherTypes [continued]

EtherType	Protocol	Notes
0x8847	MPLS (Unicast)	According to RFC 5332, this EtherType is also used for MPLS multi-cast packets when the top label was downstream assigned.
0x0800	IPv4	See Section A.5.3.1.
0x86DD	IPv6	See Section A.5.3.2.

A.5.1.3 ATA over Ethernet (AoE)

AoE is used to achieve a very basic level RPC mechanism between a client and an ATA device server. The server accepts commands and generates responses based on a command code in the AoE header.

AoE is not a connection based protocol. Each message sent to a server should be considered unique and unreliable.

The AoE packet format is illustrated in Figure A-4.

Note: Reserved fields have to be set to zero in all messages sent.

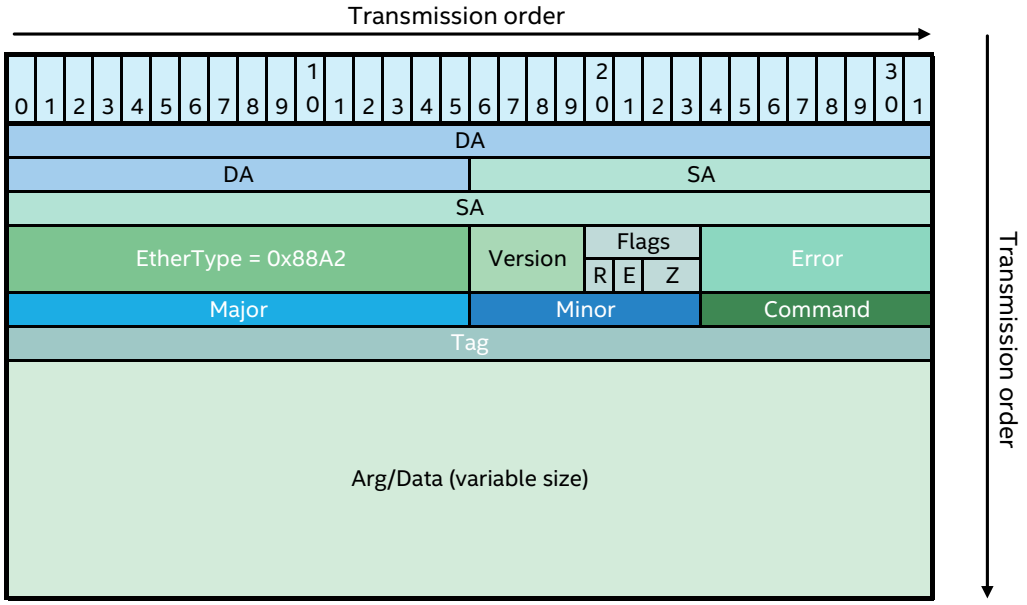


Figure A-4. AoE Frame Format

The AoE frame follows the standard Ethernet MAC header for IEEE 802.3 Ethernet.

- ***EtherType*** — 16-bit field. AoE has a registered Ethernet type of 0x88A2.
- ***Version*** — 4-bit field which defines the AoE header format and also command codes. This field must be set to 0x1.
- Flags:
  - — ***R*** — 1-bit flag that is set for a response message.
  - — ***E*** — 1-bit flag that is set for a response message if the associated command message generated an AoE protocol error.

  - ***Z*** — 2-bit field that is reserved and therefore must be set to zero.

- ***Error*** — 8-bit field. Only relevant when *E* flag is set:
  - Error 1: Unrecognized command code.
  - Error 2: Bad argument parameter.
  - Error 3: Device unavailable – The server can no longer accept ATA commands.
  - Error 4: Config string present – The server cannot set the config string because it is not empty.
  - Error 5: Unsupported version – The server does not recognize the version specified in the *Version* field.
  - Error 6: Target is reserved – Thus, the command cannot be completed.
- ***Major***, ***Minor*** — Each AoE server has a *Major* (16-bit) and a *Minor* (8-bit) address. Before processing the header *Command*, the server must validate its major and minor address with the *Major* and *Minor* fields of the header. To accept to process a command message, these conditions must be valid. The *Major* (minor) field in the header must be equal to the server *Major* (minor) address, or be a suite of ones: 0xffff (0xff). If they are not true, the command messages must be ignored by the server. The server has to supply its major and minor address in every response.
- ***Command*** — 8-bit field that contains the command code for the message. Four commands are defined for this specific version:
  - Command 0: Issue ATA Command
  - Command 1: Query Config Information
  - Command 2: MAC Mask List
  - Command 3: Reserve/Release
  - Commands codes number 240-255 (0xf0-0xff) are reserved for vendor-specific use.
- ***Tag*** — 32-bit field that provides a client with the means to correlate responses with their appropriate commands. It is copied into the response message by the server and otherwise is ignored.
- ***Arg/Data*** — Variable size field. It content serves as an input for the command code.

A.5.1.4 Link Aggregation Control Protocol (LACP)

The Link Aggregation Control Protocol (LACP) provides a standardized method to control the bundling of several ports to form a single logical channel.

The destination address is the Slow Protocols Multicast address (01-80-C2-00-00-02).

The source address in LACP PDUs is the individual MAC Address associated with the port from which the PDU is transmitted.

There are two versions of LACP frames. The first version of the LACP packet format is illustrated in Figure A-5, and the second version frame format is illustrated in Figure A-6.

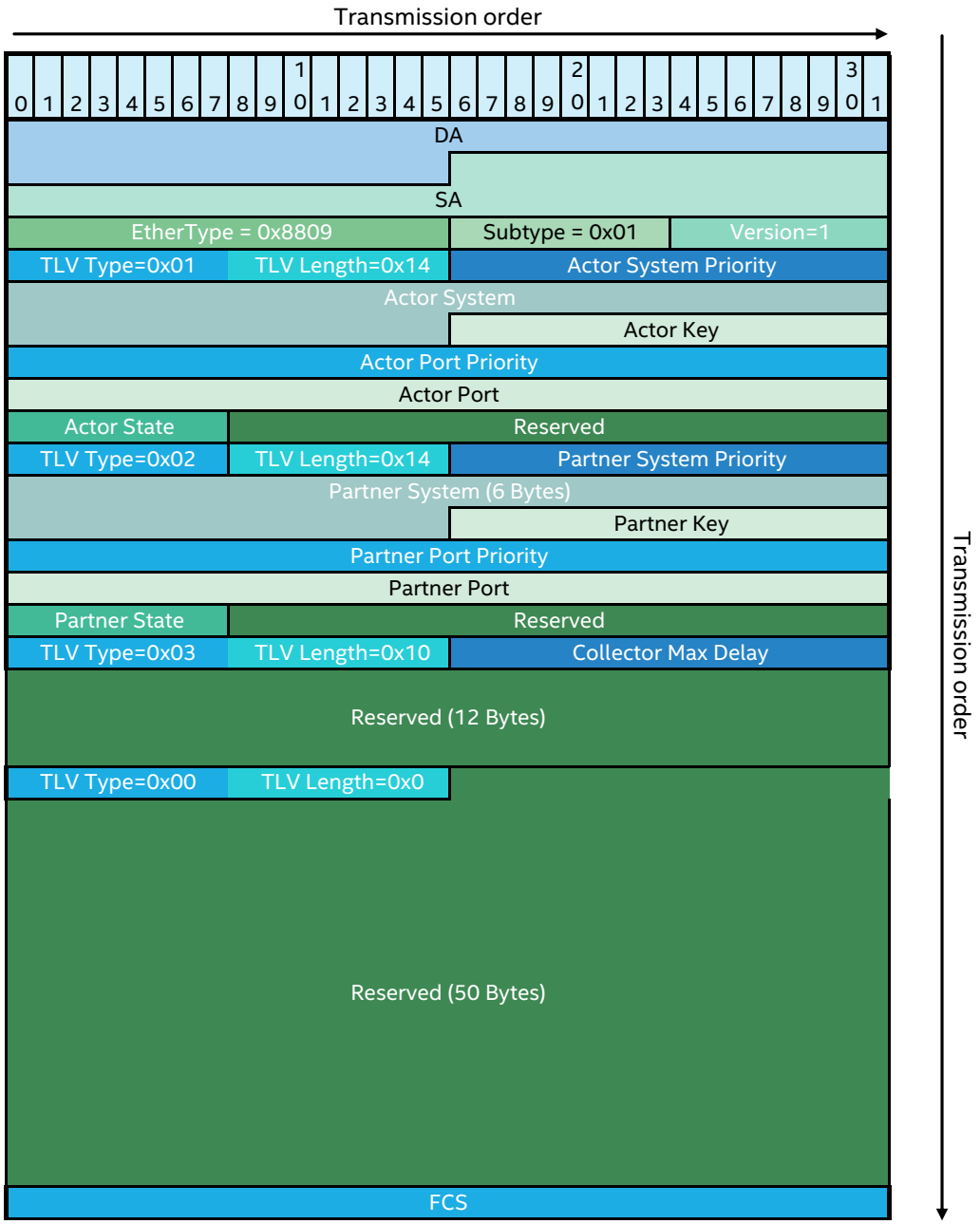


Figure A-5. LACP Version 1 Frame Format

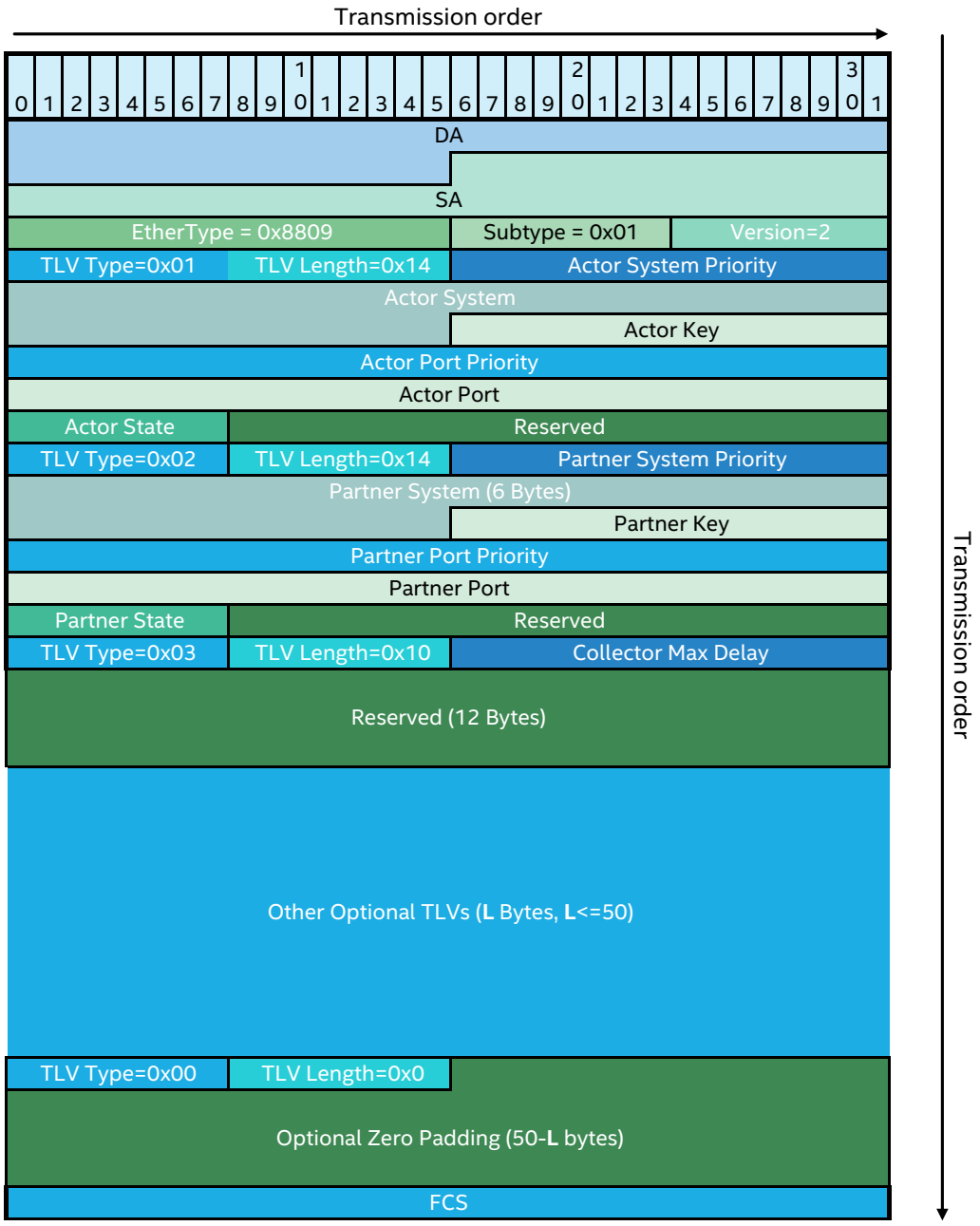


Figure A-6. LACP Version 2 Frame Format

- ***EtherType*** — The EtherType used by LACP is 0x8809.
- ***Subtype*** — 8-bit field that identifies the specific Slow Protocol being encapsulated. The subtype value is: 0x01.
- ***Version*** — 4 bit field. The first version of the LACP standard carries the value 0x01, the second one 0x02.

- ***TLV Type*** (corresponds to the Actor Information) — 8-bit field that indicates the nature of the information carried in this TLV-tuple. It is fixed to the value of 0x01.
- **TLV Length** (of the Actor information) — 8-bit field that indicates the length in bytes of the TLV-tuple. Actor information uses a length value of 20. Thus, *TLV Length* is fixed to 0x14.
- ***Actor System Priority*** — 16-bit (unsigned integer) field that indicates the priority of this System.
- ***Actor System*** — 48-bit address that indicates the MAC Address component of Actor's System ID.
- ***Actor Key*** — 16-bit (unsigned integer) field that indicates the operational key value assigned to the Aggregation Port by the actor.
- ***Actor Port Priority*** — 32-bit (unsigned integer) field that indicates the priority assigned to this Aggregation Port by the Actor (the system sending the PDU; assigned by management or administration policy).
- ***Actor Port*** — 32-bit (unsigned integer) field that indicates the port number assigned to the Aggregation Port by the Actor (the system sending the PDU).
- ***Actor State*** — 8-bit field (see Figure A-7) that defines the Actor's state variables for the Aggregation Port, encoded as individual bits within a single octet.

Figure A-7. State Variables

  - ***LACP_Activity*** — Indicates the Activity control value. For an Active LACP, the flag is set. Otherwise, the flag is cleared (0).
  - ***LACP_timeout*** — Indicates the timeout control value. For short timeout, the flag is set. Otherwise, the flag is cleared (0).
  - ***Aggregation*** — If the flag is set, the system considers this link to be Aggregable, and it is a potential candidate for aggregation. Otherwise, the link is considered to be Individual, and it can be operated only as an individual link.
  - ***Synchronization*** — If the flag is set, the system considers this link to be IN_SYNC (it has been allocated to the correct LAG, the group has been associated with a compatible Aggregator, and the identity of the LAG is consistent with the System ID and operational Key information transmitted). Otherwise, this link is currently OUT_OF_SYNC (it is not in the right LAG).
  - ***Collecting*** — If the flag is set, collection of incoming frames on this link is definitely enabled and it is not expected to be disabled in absence of administrative changes in received protocol information. Otherwise, the flag is cleared (0).
  - ***Distributing*** — If the flag is 0, the distribution of outgoing frames in this link is currently disabled and it is not expected to be enabled in absence of administrative changes or changes in received protocol information. Otherwise, the flag is set (1).
  - ***Defaulted*** — If the flag is set, it indicates that the Actor's Receive machine is using defaulted operational partner information, administratively configured for the partner. Otherwise, the operational partner information in use has been received in a LACPDU.
  - ***Expired*** — If the flag is set, it indicates that the Actor's received machine is in the EXPIRED state. Otherwise, it is not.

    Note: The received values of Defaulted and Expired State (Bits 6 and 7) are not used by LACP, but their values can be useful for diagnosing protocol problems.

- ***Reserved*** — 24-bit field that is not used in the actual protocol (it is reserved for the future extensions of the protocol). On receipt, should be ignored. On transmission, should be set to zeros.
- ***TLV Type*** (corresponds to the Partner information) — 8-bit field that indicates the nature of the information. Set to the value of 0x02.
- ***TLV Length*** (of the Partner information) — 8-bit field that indicates the length in bytes of the TLV-tuple. Actor information uses a length value of 20. Thus, *TLV Length* is fixed to 0x14.
- ***Partner System Priority*** — 16-bit (unsigned integer) field that indicates the priority of the Partner System.
- ***Partner System*** — 48-bit address that indicates the MAC Address component of Partner's System ID.
- ***Partner Key*** — 16-bit (unsigned integer) field that indicates the operational key value assigned to the Aggregation Port associated with this link by the partner.
- ***Partner Port Priority*** — 32-bit (unsigned integer) field that indicates the priority assigned to this Aggregation port by the partner.
- ***Partner Port*** — 32-bit (unsigned integer) field that indicates the port number assigned to the Aggregation Port by the Partner.
- ***Partner State*** — 8 bit field (see Figure A-7) that defines the partners state variables for the Aggregation Port, encoded as individual bits within a single octet, as defined for Actor State.
- ***Reserved*** — 24-bit field that is not used in the actual protocol (it is reserved for the future extensions of the protocol). On receipt, should be ignored. On transmission, should be set to zeros.
- ***TLV Type*** (corresponds to the Collector Information) — 8-bit field that indicates the nature of the information carried in this TLV-tuple. It is fixed to the value of 0x03.
- ***TLV Length*** (of the Collector information) — 8-bit field that indicates the length in bytes of the TLV-tuple. Actor information uses a length value of 16. Thus, *TLV Length* is fixed to 0x10.
- ***Collector Max Delay*** — 16-bit (unsigned integer) field that contains the Collector Max Delay value of the station that transmits the LACPDU. In tens of microseconds. The range of value is 0 to 63,535 tens of microseconds (0.65535s).
- ***Reserved*** — 12-byte field that is not used in the actual protocol (it is reserved for the future extensions of the protocol). On receipt, should be ignored. On transmission, should be set to zeros.
- ***Other Optional TLVs*** — Exists only for the second version of the protocol. Up to 50-byte field. the length is defined in byte by *L*.

  Note: For Long LACPDUs, *L* can be up to 1438. However, the E810 supports headers up to 504 bytes, and therefore does not support Long LACPDUs.
- ***TLV Type*** (corresponds to the Terminator Information) — 8-bit field that indicates the nature of the Terminator information (end of the message) carried in this TLV-tuple. It is fixed to the value of 0x00.
- ***TLV Length*** (of the Terminator information) — 8-bit field that indicates the length in bytes of the TLV-tuple. Terminator information uses a length value of 0. Thus, *TLV Length* is fixed to 0x00. The use of a Terminator Length of 0 is intentional.
- ***Reserved*** — 50 bytes for the first version. On receipt, should be ignored. On transmission, should be set to zeros. This field is not use for Long LACPDUs.
- ***Optional Zero Padding*** — For the second version, it is a 50–L bytes field.
- ***FCS*** — Frame Check Sequence.

A.5.1.5 Point-to-Point Protocol over Ethernet (PPPoE)

The PPPoE frame format is illustrated in Figure A-8.

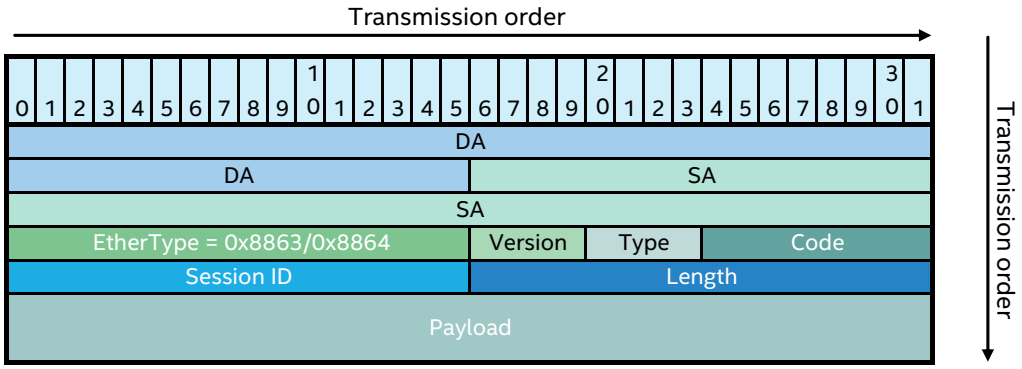


Figure A-8. PPPoE Frame Format

- ***Destination and Source Address*** — 48-bit addresses.

  The destination address contains either a unicast Ethernet destination address, or the Ethernet broadcast address (0xFFFFFFFF). For Discovery packets, the value is either a unicast or broadcast address. For PPP session traffic, this field has to contain the peer's unicast address.

  The source address has to contain the Ethernet MAC Address of the source device.

- ***EtherType*** — 16 bit field that is set to 0x8863 in the case of Discovery Stage or 0x8864 for PPP Session stage.
- ***Version*** — 4-bit field that must be set to 0x1 for this version of the PPPoE specification.
- ***Type*** — 4-bit field that must be set to 0x1 for this version of the PPPoE specification.
- ***Code*** — 8-bit field. May be set to different values (0x09, 0x07, 0x19, 0x65, 0xa7) depending on the type of Discovery packet.
- ***Session ID*** — 16-bit (unsigned) field in network byte order. Session ID takes the value of 0x0000 if the session has not yet been established. Otherwise, it is set to a unique value generated for the current session. For a given PPP session, this value is fixed. For future use, the value 0xffff is reserved.
- ***Length*** — 16-bit field in network byte order which indicates the length of the payload.
- ***Payload*** — Its length depends of the *Length* field.

A.5.1.6 Link Layer Discovery Protocol (LLDP)

The general structure of an LLDP (Link Layer Discovery Protocol) packet is depicted in Figure A-9.

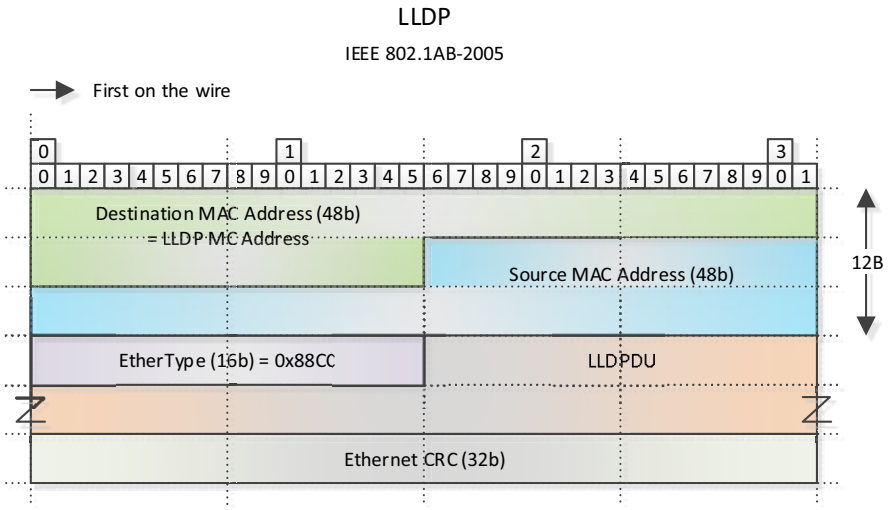


Figure A-9. Structure of LLDP Frame

The LLDPDU contains an ordered sequence of three mandatory TLVs followed by zero or more optional TLVs, plus an End of LLDPDU TLV, as shown in Figure A-10.

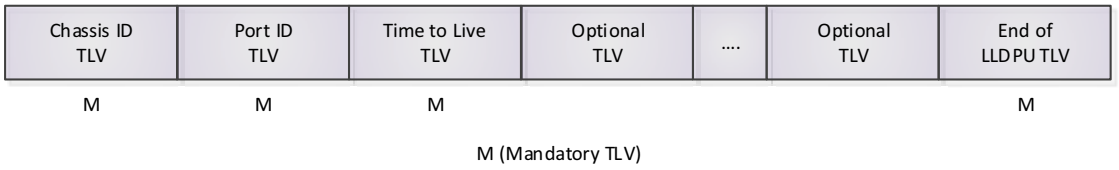


Figure A-10. Structure of LLDP PDU

Each TLV has the structure depicted in Figure A-11.

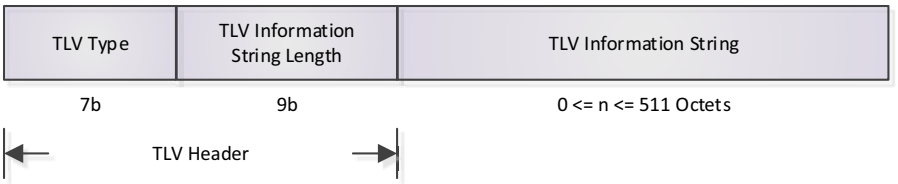


Figure A-11. Structure of an LLDP TLV

The basic format for Organizationally Specific TLVs is shown in Figure A-12.

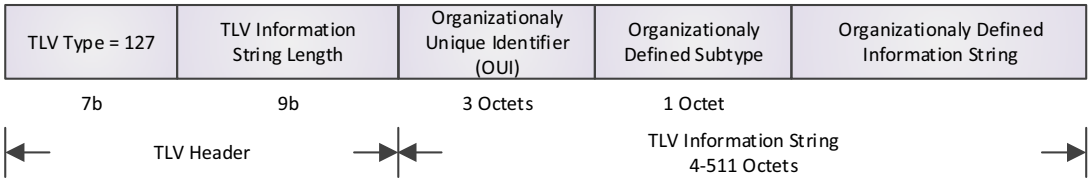


Figure A-12. Structure of LLDP Organizationally Specific TLVs

Table A-4 lists the applicable Organizationally Specific TLVs:

Table A-4. List of Organizationally Specific TLVs

TLV	OUI Value	Subtype Value
DCBX	00-80-C2	09, 0A, 0B, 0C
EEE	00-12-0F	05

Note: The E810 does not parse the TLV sections on LLDP

A.5.1.7 Magic Packets

Magic packets are described in Section 5.3.1.

A.5.1.8 Link Control Packets

Link Control packets are described in Section 3.2.1.5.1. Such packets do not include any L2 tags other than those described therein.

A.5.2 Layer 2.5

A.5.2.1 Multi-Protocol Label Switching (MPLS)

The general structure of an MPLS header is illustrated in Figure A-13.

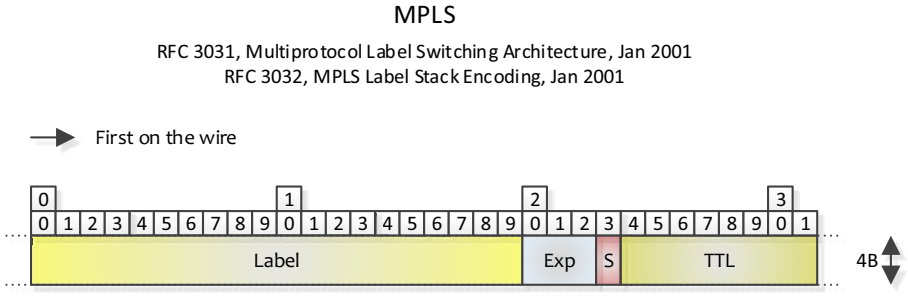


Figure A-13. Structure of MPLS Header

An MPLS header contains one or more repeating sections of:

- ***Label*** (20 bits) — Label value.
- ***Exp*** (3 bits) — QoS TC (Traffic Class) and ECN (Explicit Congestion Notification).
- ***S*** (1 bit) — Bottom-of-Stack flag, if set indicates the last label in the stack.
- ***TTL*** (8 bits) — Time-to-Live.

An MPLS header normally follows the L2 section of the frame, preceding the L3 section (the IP header). The L3 protocol following the last MPLS label is identified by the first nibble as either IPv4 (nibble equals "4") or IPv6 (nibble equals "6").

On top of outer MPLS, the E810 supports MPLSoGRE (MPLS over GRE) and MPLSoUDP (MPLS over UDP).

A.5.3 Layer 3

A.5.3.1 IPv4

The frame format of an IPv4 datagram is illustrated in Figure A-14.

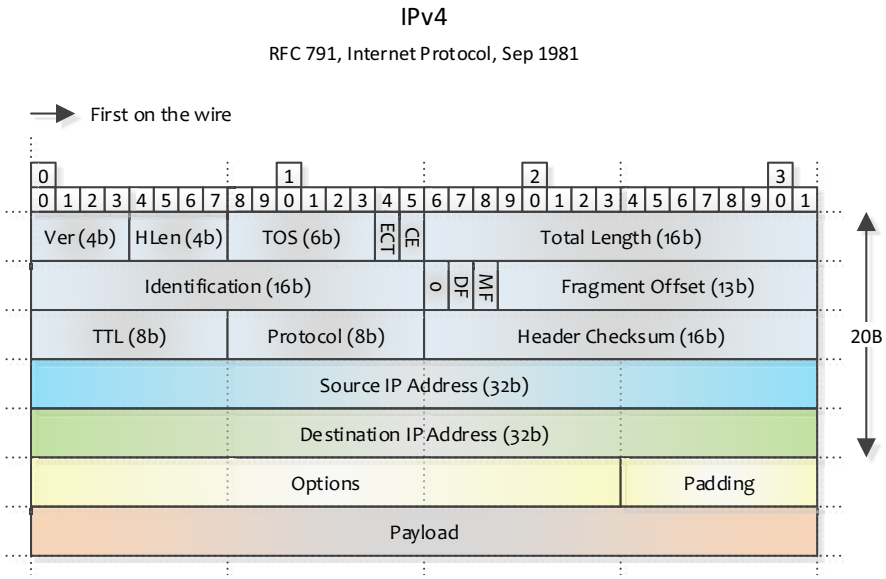


Figure A-14. IPv4 Header Format

- ***Ver*** (Version) (4 bits) — The *Version* field is set to 0x4 for IPv4 header.
- ***HLen*** (Header Length) (4 bits) — The length of the IP header defined in DWord units.
- ***TOS*** (Type of Service) (6 bits) — The *TOS* field is used to indicate the quality of service.
- IP Congestion Flags (2 bits): **CE**, **ECT**
- ***Total Length*** (16 bits) — The *Total Length* is the size of the IP datagram (IP header and payload) in byte units.

- ***Identification*** (16 bits) — The *Identification* field identifies a specific IP packet sent between a source and destination node. The sending host sets the *Identification* field's value, and the field is incremented for successive IP datagrams. The *Identification* field is used to identify multiple fragments of an original IP datagram.
- Fragment Parameters (3 bits):
  - — ***O*** (Fragment Offset) flag
  - — ***DF*** (Disable Fragmentation) flag
  - — ***MF*** (More Fragments) flag
- ***TTL*** (Time-to-Live) (8 bits)— The *TTL* field indicates the number of links that this IP datagram can travel before an IP router discards it.
- ***Protocol*** (Next Header) (8 bits) — The *Protocol* field indicates the next protocol encapsulated within the IP layer.
- ***Header Checksum*** (16 bits) — The *Header Checksum* field is a 16-bit one's complement of the one's complement sum of all 16-bit words in the IP header.
- ***Source and Destination IP Addresses*** — 2 x 32 bit IP Addresses.

Table A-5 describes the processing done to validate IPv4 packets:

Table A-5. IPv4 Packet Structure Validation

Field	Value	Action	Comment
Version/HDR Length	0x4	Check	Check IPv4.
Type of Service	-	Ignore	
Packet Length	-	Ignore	
Identification	-	Ignore	
Fragment Info	-	Ignore	
Time-to-Live	-	Ignore	
Protocol	Multiple	Compare	By the parse graph.
Header Checksum	-	Ignore	Protocol validation does not depend on the header checksum.
Source IP Address	-	Ignore	
Destination IP Address	-	Ignore	

A.5.3.2 IPv6

The frame format of an IPv6 datagram is illustrated in Figure A-15.

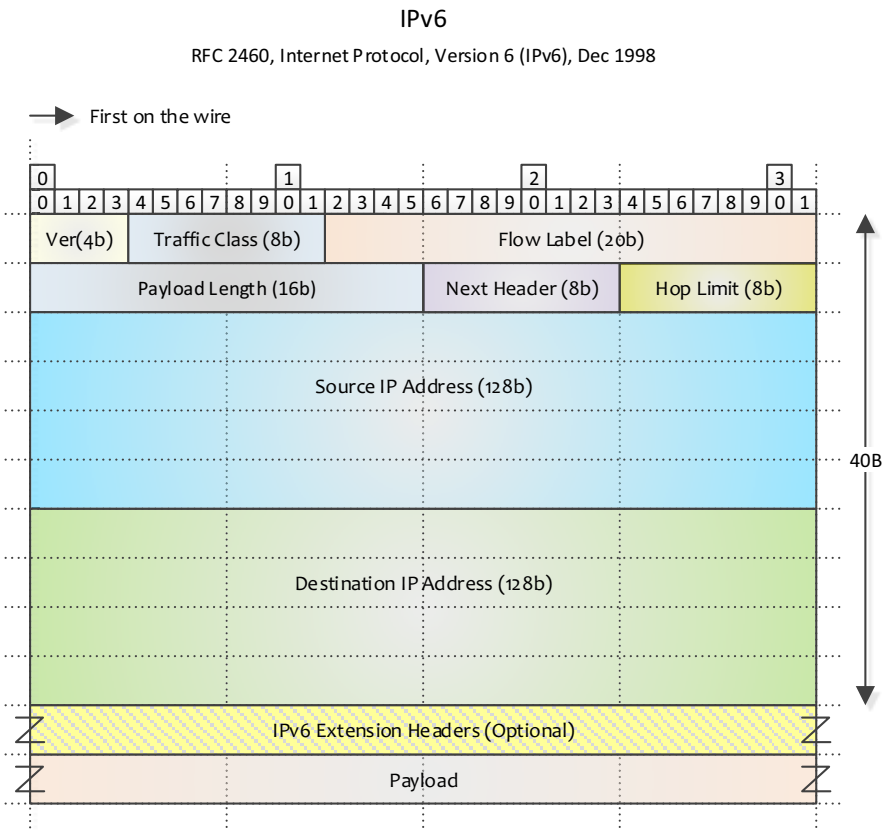


Figure A-15. IPv6 Header Format

- ***Ver*** (Version) (4 bits) — The *Version* field is set to 0x6 for IPv6 header.
- ***Traffic Class*** (8 bits)— The *Traffic Class* is used for QOS support.
- ***Flow Label*** (20 bits) — The *Flow Label* is used for QOS support.
- ***Payload Length*** (16 bits) — The length of the IPv6 payload. That is, the rest of the packet following the IPv6 header, in byte units. Note that any IPv6 extension headers are considered part of the payload.
- ***Next Header*** (Protocol) (8 bits) — The *Next Header* field indicates the next protocol encapsulated within the IP layer.
- ***Hop Limit*** (TTL) (8 bits) — The *Hop Limit* indicates the number of links that this IP datagram can travel before an IP router discards it.
- ***Source and Destination IP Addresses*** — 2 x 128 bit IP Addresses.

Table A-6 describes the processing done to validate IPv6 packets:

Table A-6. IPv6 Packet Structure Validation

Field	Value	Action	Comment
Version	0x6	Check	Check IPv6.
Traffic Class	-	Ignore	
Flow Label	-	Ignore	
Payload Length	-	Ignore	
Next Header	Multiple	Compare	By the parse graph.
Hop Limit	-	Ignore	
Source IP Address	-	Ignore	
Destination IP Address	-	Ignore	

A.5.3.3 IPv6 Extension Headers

The IPv4 option headers that are included as part of the IP header are replaced in IPv6 by separate extension headers per option. Table A-7 lists those most used IPv6 extension and their recommended ordering in the packet. Then these extension headers are illustrated in Figure A-16 through Figure A-21.

Table A-7. IPv6 Extension Headers and Their Recommended Ordering

Header (Protocol)	Next Header Value	Header Length and Header Length Field Offset
Hop-by-Hop Options	0	Variable length field defined in 8-byte units excluding the first 8 bytes.
Routing Header	43	Variable length field defined in 8-byte units excluding the first 8 bytes.
Fragment Header	44	Length is always 8 bytes. The E810 does not continue to parse the rest of the packet.
Encapsulating Security Payload	50	Length is 8 bytes plus variable length of Initial Value plus a trailer. When this header is found, the E810 does not continue to parse the rest of the packet. Applicable to IPv4 as well.
Destination Options	60	Variable length field defined in 8-byte units excluding the first 8 bytes.
Mobility Header	135	Variable length field defined in 8-byte units excluding the first 8 bytes.
No Next Header	59	When no next header type is found, the rest of the packet is not processed.

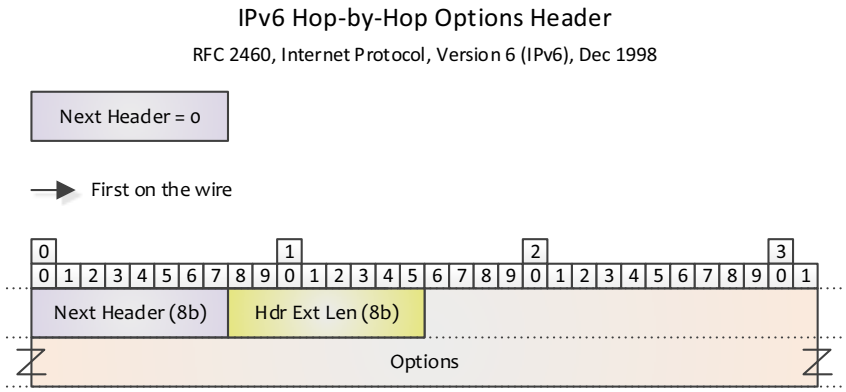


Figure A-16. IPv6 Hop-by-Hop Extension Header

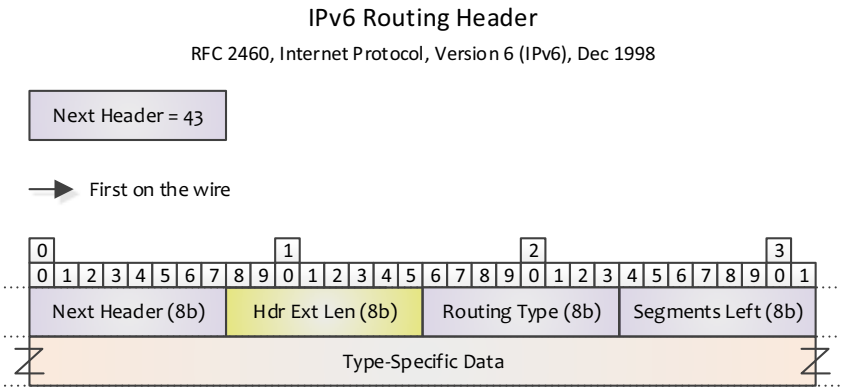


Figure A-17. IPv6 Routing Header

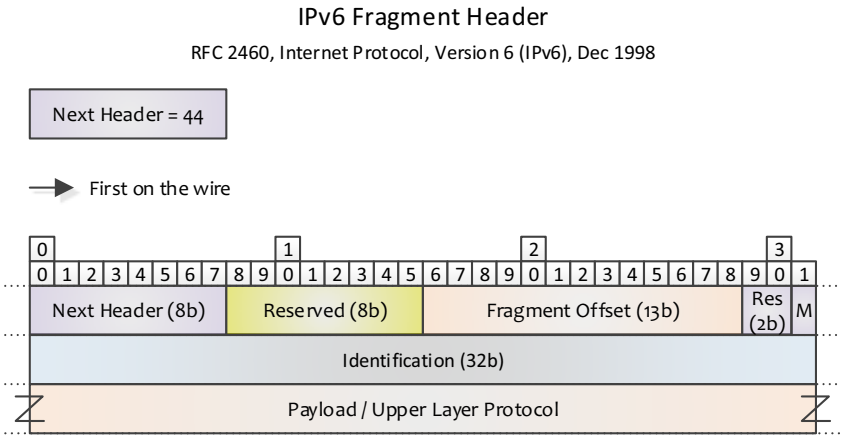


Figure A-18. IPv6 Fragment Header

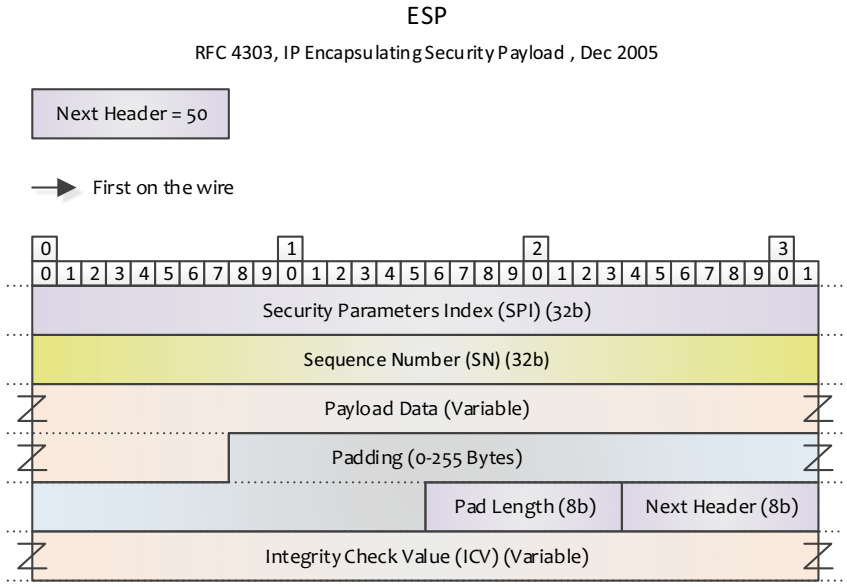


Figure A-19. Encapsulating Security Payload (ESP)

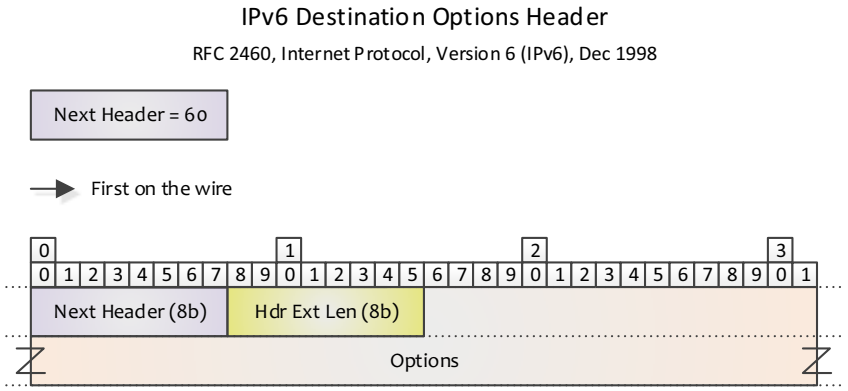


Figure A-20. IPv6 Destination Options

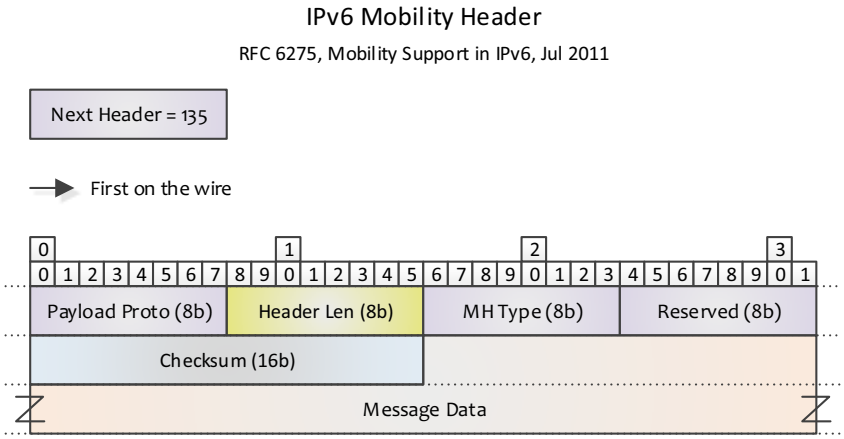


Figure A-21. IPv6 Mobility Header

A.5.3.3.1 Protocol Headers (Next Header)

Table A-8 lists the encoding of the *Next Header Type* field and information on determining each header type's length recognized by the E810.

Table A-8. Header Type Encoding and Lengths

Header (Protocol)	Next Header Value	Header Length and Header Length Field Offset
IPv4	4	Length Field is at bits[7:4], defined in 4-byte units.
IPv6	41	Length is always 40 bytes.
Encapsulating Security Payload	50	Length is 8 bytes plus variable length of Initial Value plus a trailer.
TCP	6	Length field is at Byte 12, Bits [7:4], defined in 4-byte units.
UDP	17	Length is always 8 bytes.
ICMP	1	Length is always 8 bytes.
ICMPv6	58	Length is always 4 bytes.
SCTP	132	Length is always 12 bytes.
VRRP	112	Virtual Router Redundancy Protocol ***Note:*** In Factory Parse Graph only over IPv4.
OSPFv3	89	Open Shortest Path First Version 3 (covered in RFC 5340) ***Note:*** In Factory Parse Graph only over IPv4.
No Next Header	59	When no next header type is found, the rest of the packet is not processed.

A.5.3.4 Internet Control Message Protocol (ICMP)

ICMP packets are used as part of the manageability filtering. The ICMP header structure is illustrated in Figure A-22.

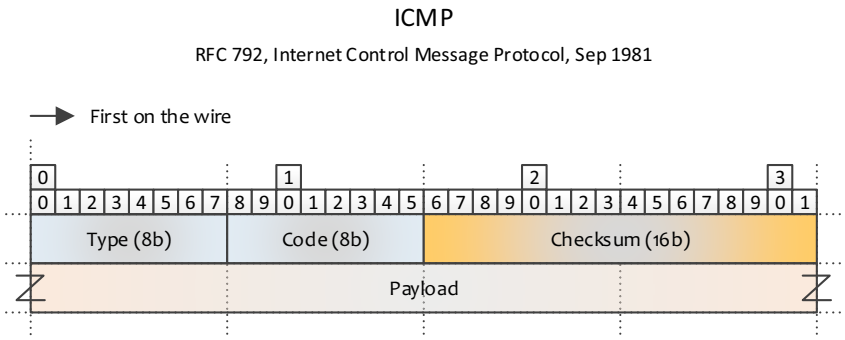


Figure A-22. ICMP Header Structure

Note: The E810 parser does not parse into the ICMP header and only detects the IPv4 *Next Protocol* as ICMP.

Table A-9. ICMP Packet Structure and Processing

Field	Value	Action	Comment
Preceding Protocol Header	IPv4	Check	Enforced by the parse graph.
ICMP type	-	Ignore	
ICMP Code	-	Ignore	
ICMP Header Checksum	-	Ignore	
ICMP Payload	-	Ignore	

A.5.3.5 ICMPv6

ICMPv6 packets are used as part of the manageability filtering and as part of proxy capabilities. The ICMPv6 header structure is illustrated in Figure A-23.

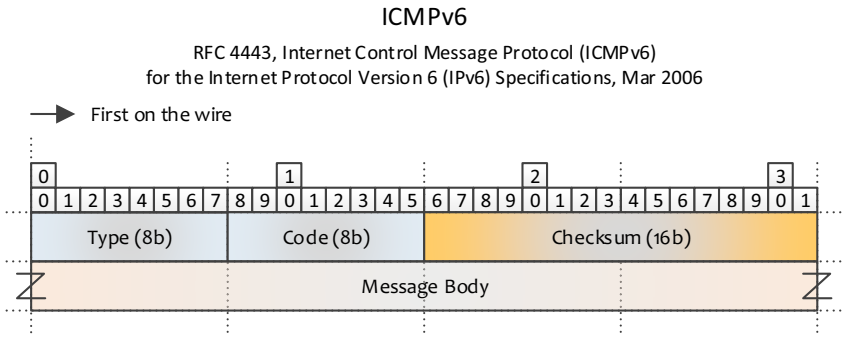


Figure A-23. ICMPv6 Header Structure

Note: The E810 parser detects ICMPv6 by evaluating the *Next Header* field following IPv6 header with optionally more Extension Headers.

The E810 supports 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

Table A-10 describes the processing done to validate ICMPv6 packets:

Table A-10. ICMPv6 Packet Structure and Processing

Field	Value	Action	Comment
Preceding L3 Protocol Header	IPv6	Check	Enforced by the parse graph.
Type	0x82, 0x83, 0x84, 0x86, 0x87, 0x88, 0x89, or 0x8F	Compare	Multicast Listener Discovery (MLD) or Neighbor Discovery types.
Code	0x0	Ignore	
Checksum		Ignore	
Message Body		Ignore	

A.5.3.6 Virtual Router Redundancy Protocol (VRRP)

Virtual Router Redundancy Protocol (VRRP), defined in RFC 5798, specifies an election protocol that dynamically assigns responsibility to one of the VRRP routers on a LAN. The "master" router controls the IP Address(es) associated with a virtual router. Forwarded packets are sent to these IP Addresses. VRRP provides a higher availability and reliability of default path without requiring configuration of dynamic routing or router discovery protocols on every end-host.

The purpose of the VRRP packet is to communicate to all VRRP routers the priority and the state of the Master router associated with the Virtual Router ID. VRRP packets are sent encapsulated in IP packets.

- The IPv4 multicast address assigned for VRRP is **224.0.0.18**.
- The IPv6 multicast address assigned for VRRP is **FF02:00:00:00:00:00:00:12**.
- The TTL must be set to **255**.

  Note: A VRRP router receiving a packet with the TTL not equal to 255 MUST discard the packet.
- The IPv4 protocol number/IPv6 Next Header protocol assigned to VRRP is **112**.

Note: The factory parsing configuration must use the next protocol as the only means of detecting a VRRP header and must not perform any validation checks on the header or preceding header related to this protocol.

The VRRP packet format is illustrated in Figure A-24.

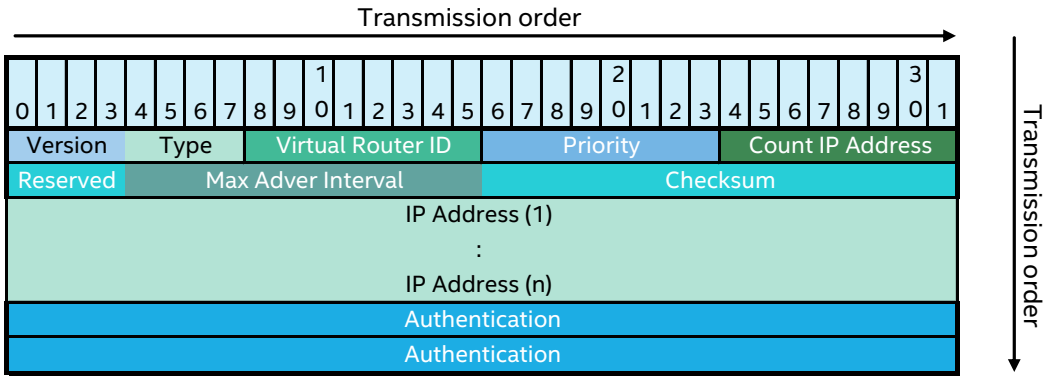


Figure A-24. VRRP Header

- ***Version*** — 4-bit field that specifies the VRRP protocol version of the packet. This specification describes version 3.
- ***Type*** — 4-bit field that specifies the type of the packet. In this version (2), only type number one (ADVERSTISEMENT) is defined. A packet with another type number has to be discarded.
- ***Virtual Router ID*** — 8-bit field that identifies the virtual router this packet is reporting status for. This is a configurable item in the range of 1-255 (decimal). There is no default value.
- ***Priority*** — 8-bit (unsigned integer) field that specifies the sending VRRP router's priority for the virtual router. The higher the value, the higher the priority. For the VRRP router that owns the IP Address(es) associated with the virtual router, priority is 255 (decimal value). This is the "Master".
  - — VRRP routers backing up a virtual router must use priority values, between 1 to 254 (decimal). The default priority value is in this case 100 (decimal).
  - — The priority value zero means that the current Master has stopped participating in VRRP. It is used to effectuate quickly the exchange of master. (We do not have to wait for the current Master to timeout).
- ***Count IP Address*** — 8-bit field that indicates the number of IP Addresses contained in this VRRP advertisement.
- ***Authentication Type*** — 8-bit (unsigned integer) field that identifies the authentication method used. Authentication type is unique on a Virtual Router basis. If a packet has an unknown authentication type or does not match the local authentication, it must be discarded.

  The authentication methods currently defined are:

  - — 0 = No Authentication — VRRP protocol exchanges are not authenticated. The *Authentication Data* field should be set to zero on transmission and ignored on reception.
  - — 1 = Reserved.
  - — 2 = Reserved.

  Types number 1 and 2 are reserved to maintain backward-compatibility with earlier version of VPPR (RFC 2338).

- ***Max Advertisement Interval*** (Adver Int) — 12-bit field that indicates the time interval (in centiseconds) between ADVERTISEMENTS. The default value is 100 centisecond (1 second). It is used for troubleshooting mis-configured routers.
- ***Checksum*** — 16-bit field that is used to detect data corruption in the VRRP message. It is the complement of complement sum of the entire VRRP message starting with the *Version* field (for computing the checksum, the *Checksum* field is set to zero).
- ***IP Address(es)*** — There are one or more IP Addresses associated with the virtual router. Each address length depends on whether it is an IPv4 or IPv6 Address. The number of IP Addresses is specified by the *Count IP Address* field. A mix of IPv4 and IPv6 Addresses is not allowed in the same VRRP header
- ***Authentication Data*** — Two strings of 32 bits each. Currently only used to maintain backward-compatibility with RFC 2338. It should be set to 0 on transmission and ignored at reception.

A.5.3.7 Open Shortest Path First (OSPF)

Open Shortest Path First (OSPF) runs directly over the IP network layer. Thus, OSPF packets are encapsulated just by IP and local data-link headers.

OSPF does not define how to fragment its packets. Rather, it depends on IP fragmentation when transmitting the packets. Therefore, it is recommended to limit the size of packets sent over virtual links to 576 bytes (if it is necessary, can be up to 65,535 bytes, including the IP header).

- One of the important feature of OSPF's IP encapsulation is the use of IP multicast. For that matter, two different IP multicast addresses are used:
  - AllSPFRouters (value address: 224.0.0.5) — Every router running OSPF should be prepared to receive packets from this address. (Hello packets, one of the five types of OSPF packets, are always sent to this destination).
  - AllDRouters (value address: 224.0.0.6) — Designated and Backup Designated Router must be prepared to receive packets destined to this address. Also, certain OSPF packets are sent to those addresses during the flooding procedure.

  Packets sent to multicast addresses are meant to travel a single hop only (IP TTL= 1).
- The IP protocol number assigned to OSPF is **89**.
- Routing protocol packets are sent with IP TOS of 0. The OSPF protocol supports TOS-based routing. All OSPF routing protocol packets are sent using the normal service TOS value of binary 0000.
- OSPF protocol packets should be given precedence over regular IP data traffic (in sending and receiving). Setting the *IP Precedence* field in the IP header to *Internetwork Control* helps implement this objective.

There are five types of OSPF packets. All of them starts start with a common 24-byte header.

The OSPF header format is illustrated in Figure A-25.

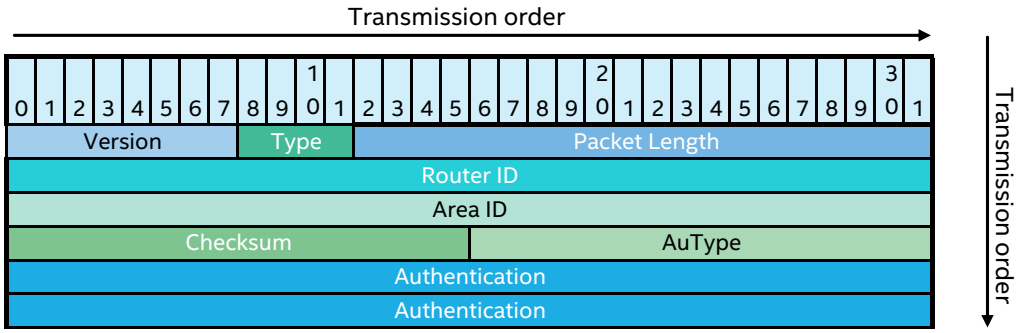


Figure A-25. OSPF Header Format

- ***Version*** — 8-bit field that indicates the version of the protocol. This section describes version 2, which is the only one supported in the factory configuration and is only relevant for IPv4.
- ***Type*** — The type of the OSPF packet: number 1 to 5.
  - — 1 = Hello
  - — 2 = Database Description
  - — 3 = Link State Request
  - — 4 = Link State Update
  - — 5 = Link State Acknowledgment
- ***Packet Length*** — 20-bits field used for the length in bytes of the packet. It includes the standard the standard OSPF header.
- ***Router ID*** — 32-bit field that indicates the Router ID of the packet's source. It is important because in OSPF, source and destination of a routing packet are the two ends of an (potential) adjacency.
- ***Area ID*** — 32 -bit field identifying the area that this packet belongs to. OSPF are associated with a single area. If packets are traveling over a virtual link, the area ID is 0.0.0.0.
- ***Checksum*** — 16-bit that indicates the standard IP checksum of the all packets. It starts with the OSPF packet header and excludes the 64-bit *Authentication* field. It is calculated as the complement of the complement sum of the 16-bit words in the packet (except authentication field). The packet is padded with a zero byte if the packet's length is not an integral number of 16-bit words before summing.
- ***AuType*** — 16-bit field that indicates the authentication scheme.
- ***Authentication*** — 64-bit field used by the authentication scheme.

A.5.3.8 Address Resolution Protocol (ARP)

ARP packets are used as part of the manageability filtering and as part of the proxy capabilities.

Figure A-26 describes the ARP packets structure.

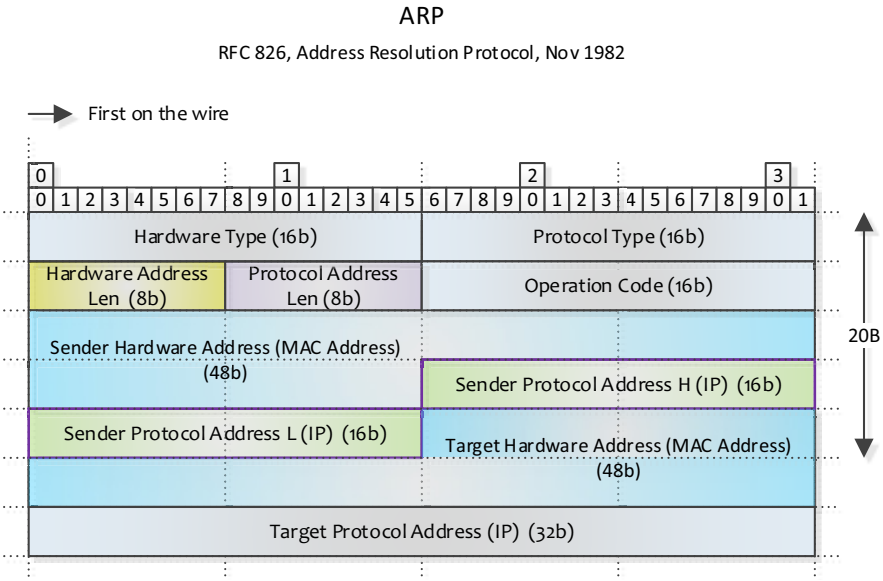


Figure A-26. ARP Packet Structure

Table A-11. ARP Packet Structure and Processing

Field	Value	Action	Comment
Preceding EtherType	0x0806	Compare	ARP.
Hardware Type	0x0001	Compare (by firmware)	Ethernet Hardware type.
Protocol Type	0x0800	Compare (by firmware)	IPv4 Protocol.
Hardware Size	0x06	Compare (by firmware)	MAC Address size in bytes.
Protocol Address Length	0x04	Compare (by firmware)	IPv4 Address size in bytes.
Operation	0x0001/0x0002	Compare (by firmware)	0x0001 = Request 0x0002 = Response
Sender Hardware Address	-	Ignore	
Sender IP Address	-	Ignore	
Target Hardware Address	-	Ignore	
Target IP Address	IP4AT	Compare (by firmware)	Use to decide of an IP match of ARP packets.

Note: The Parser identifies ARP by its ETYPE. Further validation is performed by firmware when processing the ARP packet.

A.5.4 Layer 4

A.5.4.1 User Datagram Protocol (UDP)

The UDP header structure is illustrated in Figure A-27.

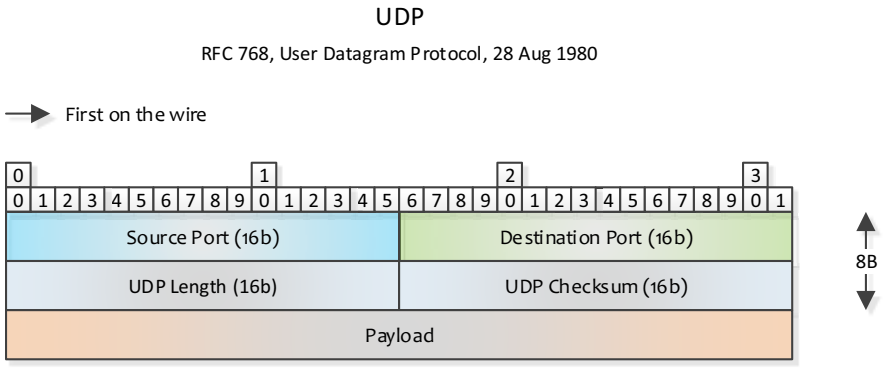


Figure A-27. UDP Packet Format

- ***Source Port*** (16 bits) — Source port number.
- ***Destination Port*** (16 bits) — Destination port number.
- ***UDP Length*** (16 bits) — The length of the entire UDP datagram, including both header and Data fields defined in byte units.
- ***UDP Checksum*** (16 bits) — An optional one's complement of the one's complement sum of all 16-bit words in the header and payload and a "pseudo header" illustrated in the figures above.

On the transmit flow, the OS stack provides the "pseudo header" checksum in the *UDP Checksum* field when requesting from the NIC to offload the checksum calculation.

Table A-12. UDP Header Structure

Field	Value	Action	Comment
Source UDP Port	XXXX	Ignore	Source UDP port number could be any value.
Destination UDP Port	VXLAN Port	Compare	Per the list of supported protocols over UDP.
UDP Length	XXXX	Ignore	Length of the entire datagram including the UDP header.
UDP Checksum	XXXX	Check	Provide an indication if the checksum equals to zero (i.e., no checksum).

Table A-13 lists the factory preset UDP Destination Port values.

Table A-13. Factory Preset UDP Destination Port Values

UDP Destination Port	Protocol	Notes
4789	VXLAN	
4790	VXLAN-gpe	
6081	Geneve	
2152	GTPu (GTPv1-U)	

Table A-13. Factory Preset UDP Destination Port Values [continued]

UDP Destination Port	Protocol	Notes
4791	RoCEv2	IANA assigned The factory parsing program supports up to two configurable UDP ports per physical port for identifying RoCEv2 packets. The value mentioned in this table is the IANA assigned port but the software might use an additional and/or different number (for example, 1021 - the legacy RoCEv2 port).
6635	MPLS over UDP	Covered in RFC 7510

A.5.4.1.1 UDP Pseudo-Header

The UDP pseudo-header structure is illustrated in Figure A-28 and Figure A-29.

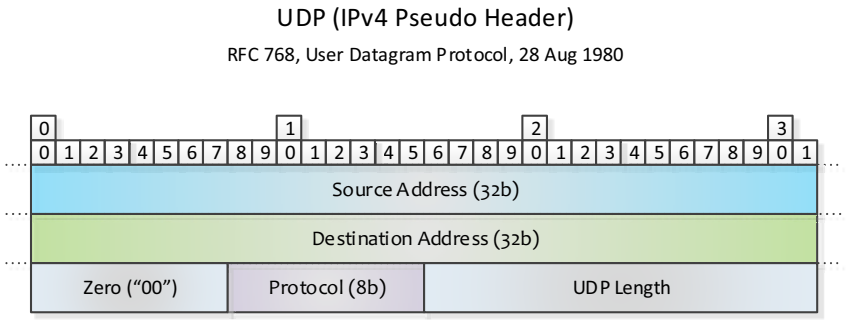


Figure A-28. IPv4 Pseudo-Header for UDP Checksum

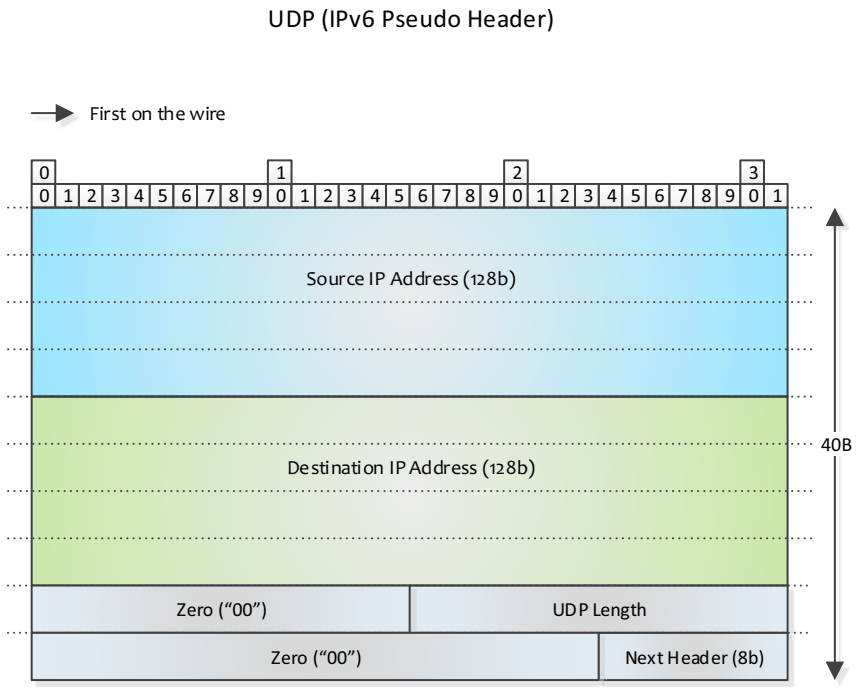


Figure A-29. IPv6 Pseudo-Header for UDP Checksum

A.5.4.2 Transmission Control Protocol (TCP)

The TCP header structure is illustrated in Figure A-30.

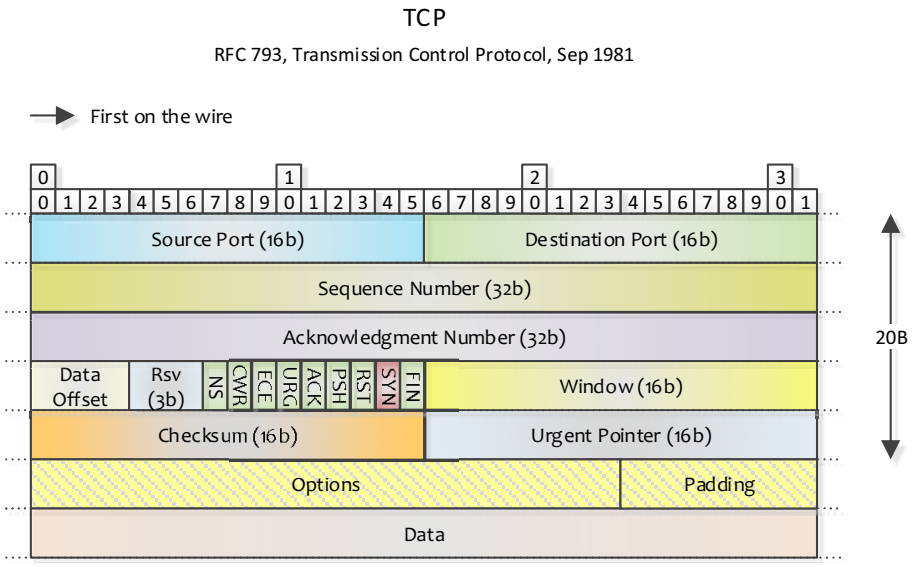


Figure A-30. TCP Packet Format

- ***Source Port*** (16 bits) — Source port number.
- ***Destination Port*** (16 bits) — Destination port number.
- ***Sequence Number*** (32 bits) — The sequence number of the first data octet in this segment (except when SYN is present). If SYN is present the sequence number is the initial sequence number (ISN) and the first data octet is ISN+1.
- ***Acknowledgment Number*** (32 bits) — If the *ACK* control bit is set, this field contains the value of the next sequence number the sender of the segment is expecting to receive. Once a connection is established this is always sent.
- ***Window Size*** (16 bits) — The number of data octets beginning with the one indicated in the *Acknowledgment* field which the sender of this segment is willing to accept.
- TCP Flags (9 bits) — **FIN**, **SYN**, **RST**, **PSH**, **ACK**, **URG**, **ECE**, **CWR**, **NS**.
- ***Header Length*** (4 bits) — The number of DWords in the TCP Header. This indicates where the data begins. The TCP header (including TCP options) is always an integral number of 32 bits long.
- ***Urgent Pointer*** (16 bits) — This field communicates the current value of the urgent pointer as a positive offset from the sequence number in this segment. The *Urgent Pointer* points to the sequence number of the octet following the urgent data. This field is only be interpreted in segments with the *URG* control bit set.
- ***TCP Checksum*** (16 bits) — The *Checksum* field is the 16-bit one's complement of the one's complement sum of all 16-bit words in the header and payload and the "pseudo header" illustrated in the figures above.

  Note: On the transmit flow, the OS stack provides the "pseudo header" checksum in the TCP *Checksum* field when requesting from the NIC to offload the checksum calculation.

A.5.4.2.1 TCP Pseudo-Header

The TCP pseudo-header structure is illustrated in Figure A-31 and Figure A-32.

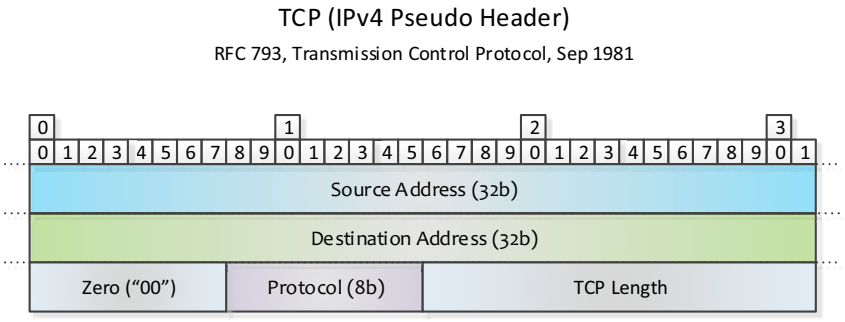


Figure A-31. IPv4 Pseudo-Header for TCP Checksum

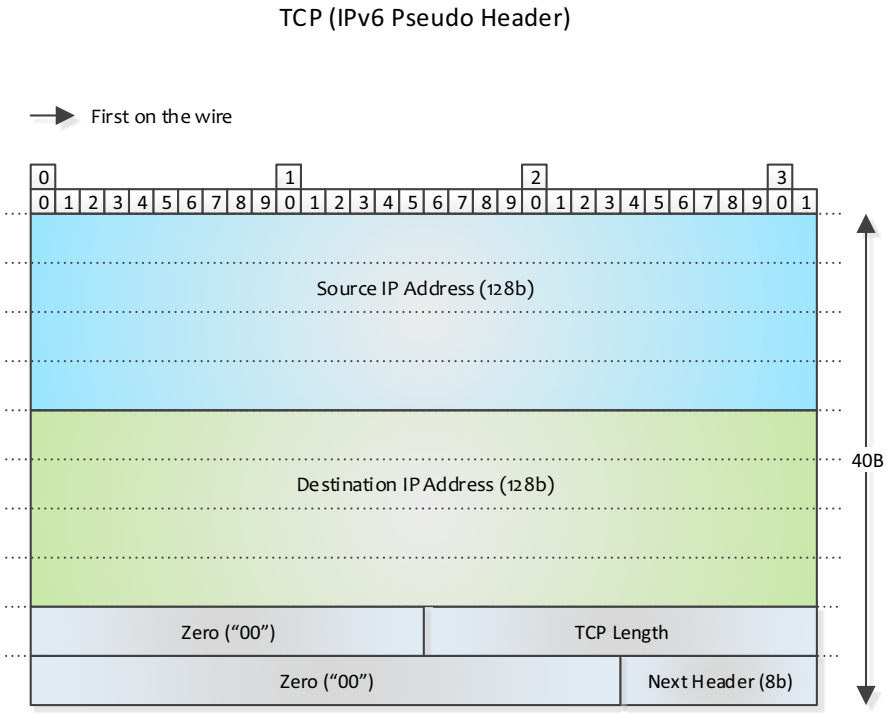


Figure A-32. IPv6 Pseudo-Header for TCP Checksum

A.5.4.3 Stream Control Transmission Protocol (SCTP)

The SCTP header structure is illustrated in Figure A-33.

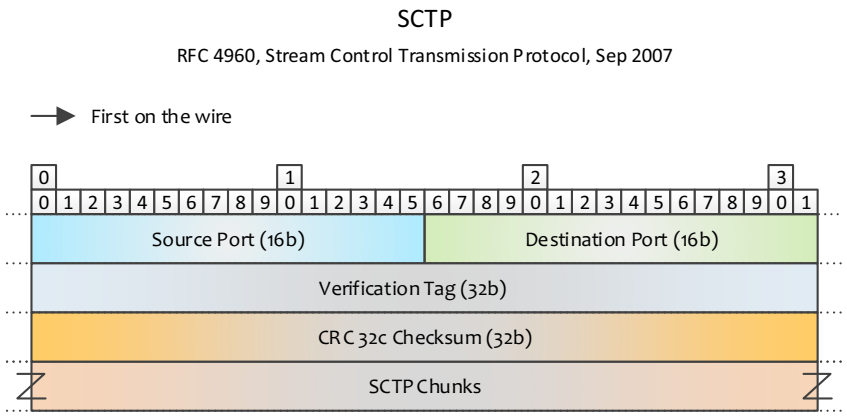


Figure A-33. SCTP Packet Structure

- ***Source Port*** (16 bits) — Source port number.
- ***Destination Port*** (16 bits) — Destination port number.
- ***Verification Tag*** (32 bits) — Random value selected by each endpoint in an association during setup. It is used to discriminate between two successive associations, as well as a protection mechanism against blind attackers.
- ***CRC Integrity*** — CRC32c integrity checksum covering the entire SCTP packet (SCTP header and all chunks). the CRC32c is the same polynomial used for iSCSI as follow: $1 + x + x^2 + x^4 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{12} + x^{16} + x^{22} + x^{23} + x^{26} + x^{32}$. The CRC bytes are transmitted on the network in big endian ordering while the MS bytes is first on the wire.

A.5.5 Tunneling and Overlay Networks

The factory parsing program in the E810 supports the following tunneling and overlay networks formats:

- MAC-in-MAC
- IP-in-IP: IPv4/IPv4, IPv4/IPv6, IPv6/IPv4, IPv6/IPv6
- NSH (Service Chaining): NSHoE, NSHoGRE, NSHoVXLANgpe
- VXLAN (MAC-in-UDP), VXLAN-gpe
- GRE
- Geneve
- MPLSoGRE, MPLSoUDP
- IPSEC NAT-T
- GTP

A.5.5.1 Generic Routing Encapsulation (GRE)

The GRE header structure is illustrated in Figure A-34.

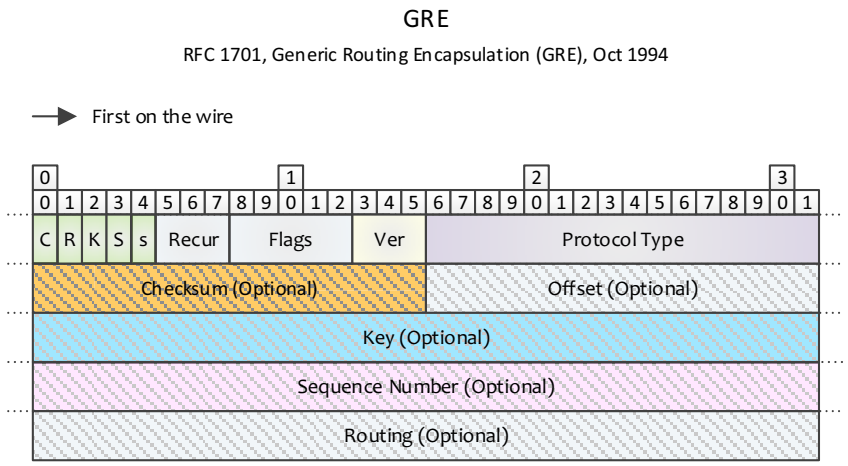


Figure A-34. GRE Header Structure (RFC 1701)

The GRE header is indicated by IP protocol equals to 47 (0x2F). The GRE headers supported by the E810 can be 1, 2, 3 or 4 DWords depending on the flags in the first byte.

- ***C*** (Checksum Present) (1 bit) — When set, it indicates that the *Checksum* field is present and contains valid information. If either the *Checksum Present* bit or the *Routing Present* bit are set, the *Checksum* and *Offset* fields are both present.
- ***R*** (Routing Present) (1 bit) — GRE header with routing header is not supported by the E810. If this flag is found active the header is not recognized by the device.
- ***K*** (Key Present) (1 bit) — If set, the *Key* field is present and contains valid information.
- ***S*** (Sequence Number) (1 bit) — If set then the *Sequence Number* field is present and contains valid information. The E810 ignores this flag other than an indication for the length of this header.
- ***s*** (Strict Source Route) (1 bit) — It is recommended that this bit only be set if all of the Routing Information consists of Strict Source Routes. The E810 ignores this flag.
- ***Recur*** (Recursion Control) (3 bits) — Contains the number of additional encapsulations that are permitted. The E810 supports only GRE header with no recursion headers (Recursion Control equal to zero).
- ***Ver*** (Version) (3 bits) — GRE protocol version. Zero value identifies a GRE header version zero. The E810 supports only zero value. (Note that version 1 is used for PPP protocol.)
- ***Protocol Type*** (16 bits) — Contains the protocol type of the payload packet. In general, the value is the *Ethernet Protocol Type* field for the packet. The following values are supported by the E810. Packet parsing that might have other protocol values is undefined.
  - — IPv4 0x0800
  - — IPv6 0x86DD
  - — MPLS 0x8847 (unicast MPLS)
  - — MAC 0x6558 (MAC-in-GRE)

— NSHoGRE 0x894F

- ***Checksum*** (16 bits) — Contains the IP (one's complement) checksum of the GRE header and the payload packet. This field is not processed by the E810.
- ***Offset*** (16 bits) — Indicates the byte offset from the start of the *Routing* field to the first byte of the active Source Route Entry to be examined. This field is not processed by the E810.
- ***Key*** (32 bits) — Contains a number that was inserted by the encapsulator. It can be used by the receiver to authenticate the source of the packet. The GRE Key is used for VSI classification if enabled by the switch filters.
- ***Sequence Number*** (32 bits) — Contains a number that is inserted by the encapsulator. It can be used by the receiver to establish the order in which packets have been transmitted from the encapsulator to the receiver. This field is not processed by the E810.
- ***Routing*** (variable length) — This field is a list of SREs and is not supported by the E810.

A.5.5.2 Virtual Extensible Local Area Network (VXLAN)

The VXLAN header format is illustrated in Figure A-35.

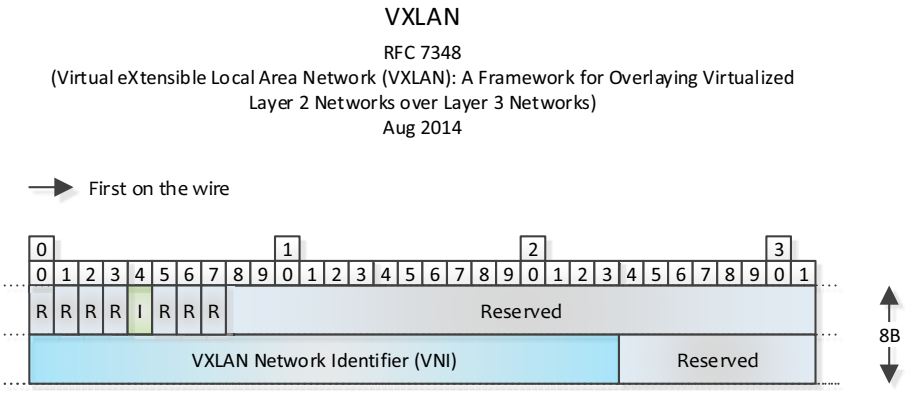


Figure A-35. VXLAN Header Structure

VXLAN comes on top of a UDP header with reserved destination port = 4789 (0x12B5). The E810 allows programming of the VXLAN port.

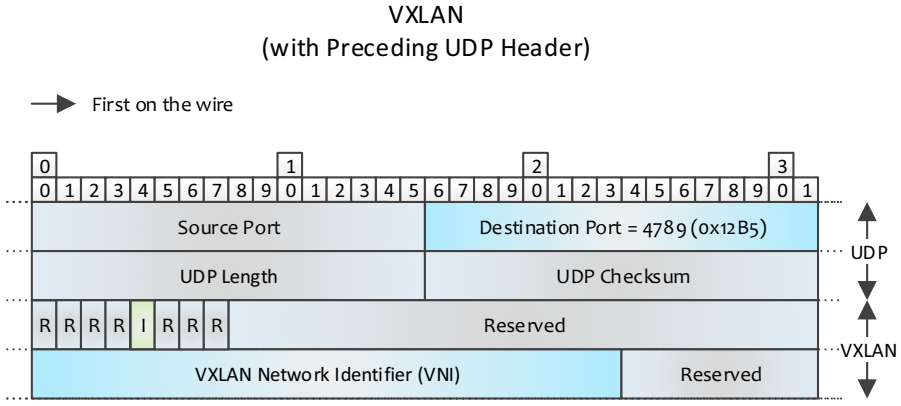


Figure A-36. VXLAN over UDP

Table A-14. VXLAN Header Structure

Field	Value	Action	Comment
Destination UDP Port	VXLAN Port	Compare	The reserved VXLAN port number is programmed by the Add "Tunneling UDP" admin command.
Flags	0x08	Ignore	I = VNI valid indication.
Reserved	XXXX	Ignore	Reserved.
VXLAN Network Identifier	XXXX	Compare	Tenant identifier (used in classification).

A.5.5.3 Generic Protocol Extension for VXLAN (VXLAN-GPE)

The VXLAN-GPE header format is illustrated in Figure A-37.

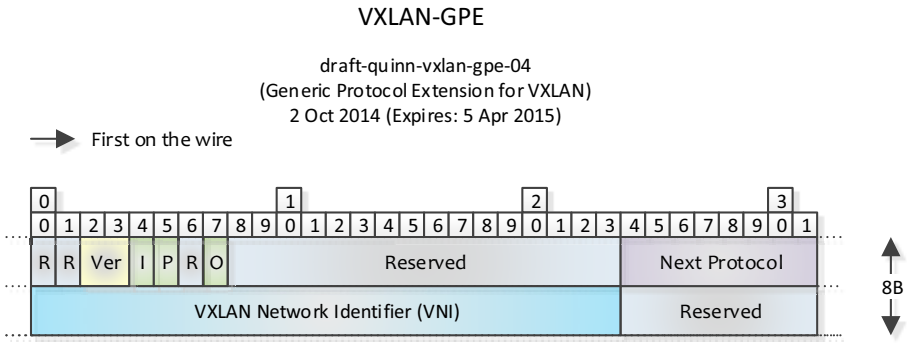


Figure A-37. VXLAN-GPE Header Structure

VXLAN-GPE comes on top of a UDP header with reserved destination port = 4790 (0x12B6). The E810 allows programming of the VXLAN-GPE port.

VXLAN-GPE
(with Preceding UDP Header)

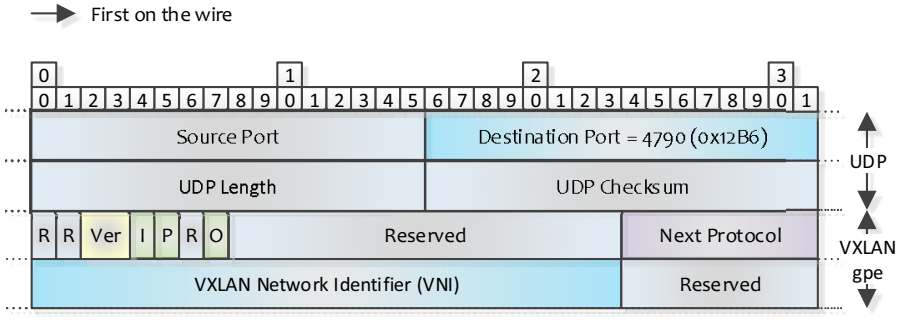


Figure A-38. VXLAN-GPE over UDP

Table A-15. VXLAN-GPE Header Structure

Field	Value	Action	Comment
Destination UDP Port	VXLAN-GPE Port	Compare	The reserved VXLAN port number is programmed by the Add "Tunneling UDP" admin command.
Flags	Ver=0 I=1 P=1 O=Ignore	Compare	Ver = Version = 0 I (Instance) = VNI valid indication = 1 P (Next Protocol) = 1 O (OAM) = Ignore
Reserved	XXXX	Ignore	Reserved.
Next Protocol	XXXX	Check	Next Protocol (according to IETF draft vxlan-gpe-03) 0x1 = IPv4 0x2 = IPv6 0x3 = Ethernet 0x4 = NSH 0x5 = MPLS
VXLAN Network Identifier	XXXX	Compare	Tenant identifier (used in classification).

A.5.5.4 Generic Network Virtualization Encapsulation (Geneve)

The Geneve header format is illustrated in Figure A-39.

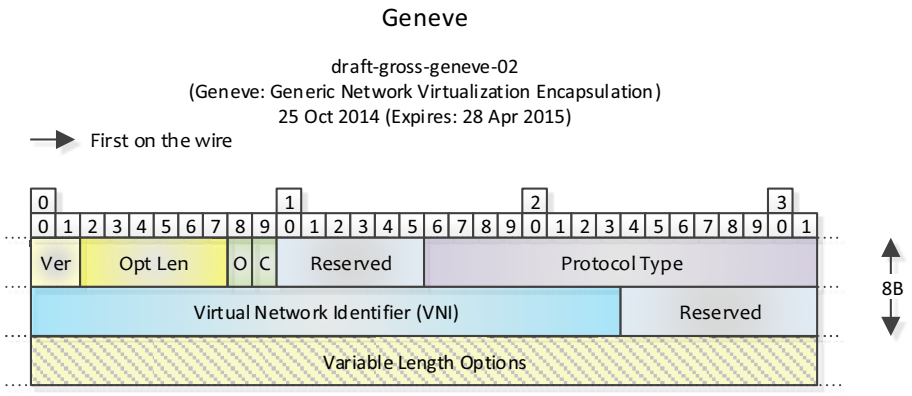


Figure A-39. Geneve Header Structure

Geneve comes on top of a UDP header with destination port = 6081 (0x17C1).

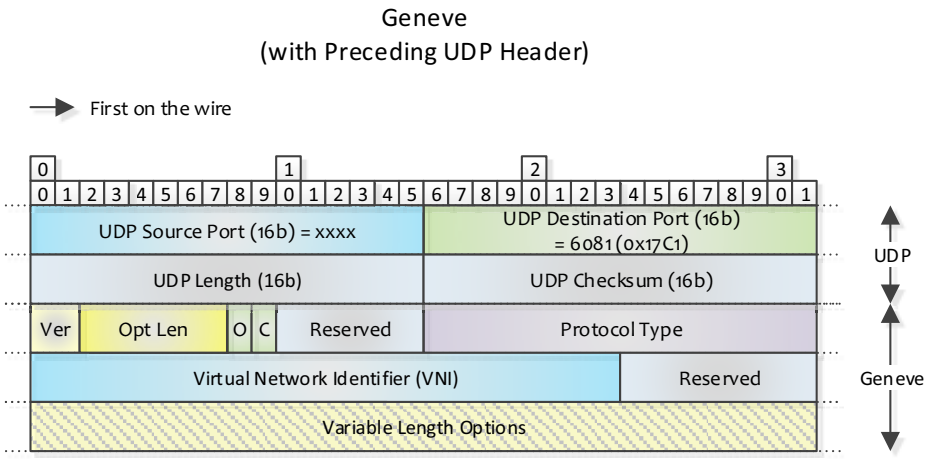


Figure A-40. Geneve over UDP

Table A-16. Geneve Header Structure

Field	Value	Action	Comment
Source UDP Port	XXXX	Ignore	Source UDP port number could be any value.
Destination UDP Port	Geneve Port	Compare	Geneve port number.
UDP Length	xxxx	Ignore	Length of the entire datagram including the UDP header.
UDP Checksum	xxxx	Check	Checksum integrity indication.
Version	00b	Ignore	Geneve version.
O (OAM Frame)	0b	Ignore	
Option Length	Variable	Check	Defines the length of the options fields in 4 byte units. The Parser is using the option length to calculate the total Geneve header length.
C (Critical Options Present)	Variable	Ignore	Expected to be processed by the software stack.
Protocol Type	Variable	Check	Next protocol.
VNI	Variable	Compare	Tenant identifier (used in classification).
Variable Length Options	Variable	Ignore	Length of the *Options* field is defined by the *Option Length* field in this header.

A.5.5.5 RDMA over Converged Ethernet v2 (RoCEv2)

The RoCEv2 packet format is illustrated in Figure A-41.

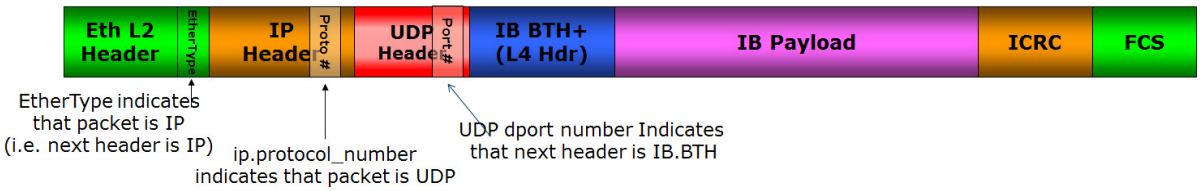


Figure A-41. RoCEv2 Packet Format

The E810 identifies RoCEv2 packets based on their UDP destination port number. Two UDP port numbers are provided per each of the physical Ethernet ports).

A.5.5.6 Network Service Header (NSH)

A Network Service Header (NSH) is used to create network service paths. In addition to path information, this header also carries metadata used by network devices and/or network services.

The NSH basic header structure is shown in Figure A-42.

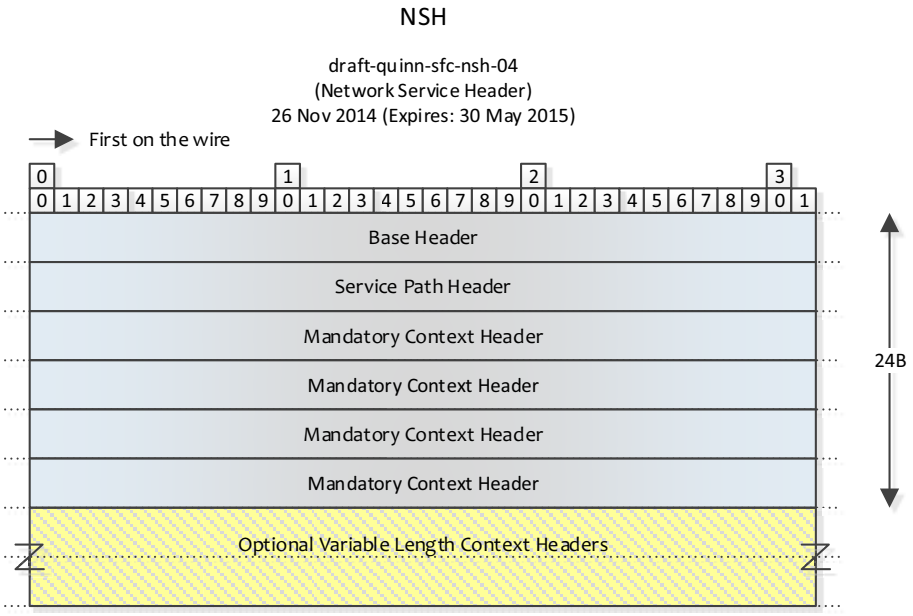


Figure A-42. NSH Basic Header Structure

The NSH header is composed of the following fields:

- ***Base Header*** — Provides information about the service header and service path identification:
  - — Version.
  - — O bit – Indicates that this packet is an operations and management (OAM) packet.
  - — C bit – Context headers MUST be present. When *C* is set, one or more contexts are in use (that is, a value placed in a context is significant). The *C* bit specifies that their ordering and sizing is as specified:
    - Network Platform (32 bits)
    - Network Shared (32 bits)
    - Service Platform (32 bits)
    - Service Shared (32 bits).

    A *C* bit equal to zero indicates that no contexts are in use (although they MUST be present to ensure a fixed size header) and that they can be ignored. If a context header is not in use, the value of that context header MUST be zero.
  - — R bit – All other flag fields are reserved.
  - — Protocol - Indicates the next protocol:
    - 0 = Reserved

  - 1 = IPv4
  - 2 = IPv6
  - 3 = Ethernet
  - 4..253 = Unassigned
- ***Service Index*** — TTL functionality and location within the service path. Service index MUST be decremented by service nodes after performing required services. MAY be used in conjunction with service path for path selection.
- ***Service Path*** — Identifies a service path. Participating node MUST use this identifier for path selection.
- ***Context Headers*** — Carry opaque metadata.
  — Network platform context – Provides platform-specific metadata shared between network nodes.
  — Network shared context – Metadata relevant to any network node such as the result of edge classification.
  — Service platform context – provides service platform specific metadata shared between service functions.
  — Service shared context – Metadata relevant to, and shared, between service functions.
- **TLV section** — Additional variable-size metadata.

Detailed NSH header structure is shown in Figure A-43.

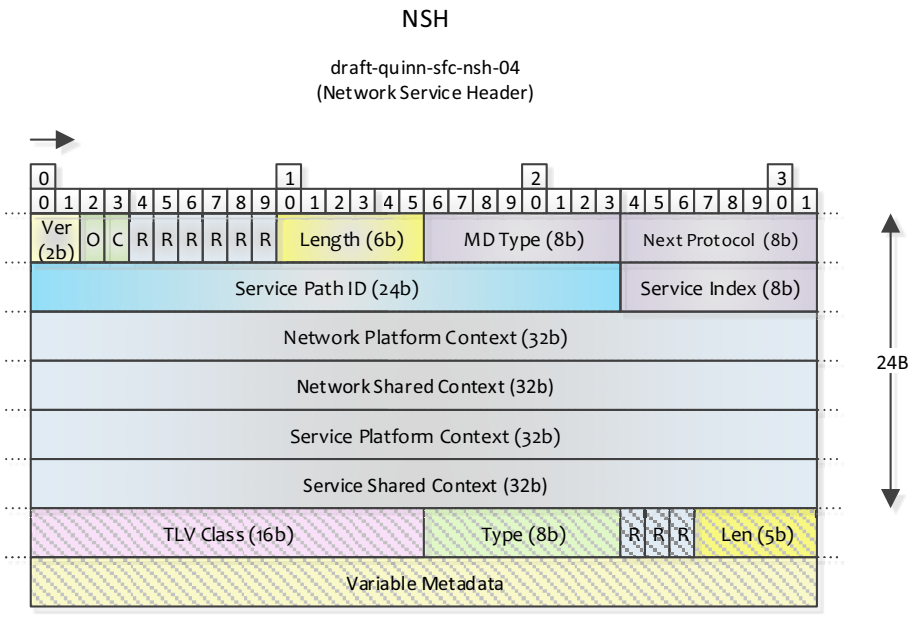


Figure A-43. Detailed NSH Header Structure

Note: The E810 supports only fixed-length NSHoE in the outer L2 section of the packet. Variable length NSH header with TLV extensions is not supported in this section.

A.5.5.7 IPSEC NAT-T

IPSEC NAT-T is identified by 1 of 8 programmable UDP destination port numbers.

The E810 does not provide checksum offload for the NAT header.

The ESP NAT-T header structure is shown in Figure A-44.

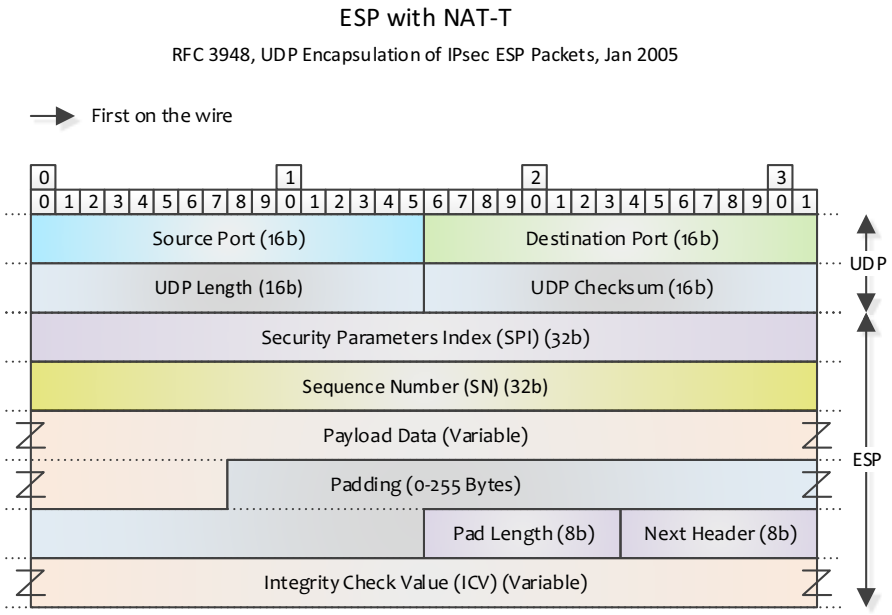


Figure A-44. ESP NAT-T Header Structure

A.5.5.8 GPRS Tunneling Protocol (GTP)

The GPRS Tunneling Protocol is specified by 3GPP.

The E810 factory parsing program supports two versions of the GTP header: GTPv1 and GTPv2, described in sections Section A.5.5.8.1 and Section A.5.5.8.2, respectively.

The protocol is transferred over IP/UDP (see Section A.5.4.1) and has distinct flavors:

- GTP-C packets, used for control, and are identified by UDP port 2123.
- GTP-U packets, used for user data, and are identified by UDP port 2152.

Note: The mentioned UDP port number must be used as the destination port for request messages and as the source port for response messages. There is no GTP-U flavor for GTPv2.

For GTP-U packets, the E810 factory parsing program also identifies whether the packet carries a user data message (G-PDU) or not.

Note: The support provided by the factory parsing program of the E810 for GTP provides the ability of detecting the GTP headers supported in accordance to the overlaying protocols and the *Version* field in the header itself (see each detailed header format sections for details). The E810 does not provide any validity check for the headers themselves, and it is the relevant software client's responsibility to implement the protocol validity checks and relevant error handling suite as mandated by the specifications.

A.5.5.8.1 GTPv1

GTPv1 is specified in 3GPP TS 29.060, and the information below is based on version 14.2.0 published in December, 2016.

The frame format for GTPv1 is described in Figure A-45.

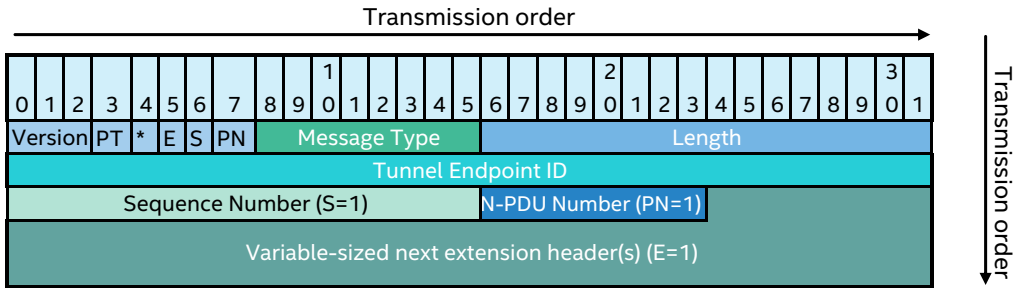


Figure A-45. GTPv1 Header Format

The following fields are always present fields in GTPv1 header:

- ***Version*** — This field is used to determine the version of the GTP protocol. The version number must be set to "1".
- ***PT*** (Protocol Type) — This bit is used as a protocol discriminator between GTP (when PT is "1") and GTP' (when PT is "0"). GTP is described in 3GPP TS 29.060 and the GTP' protocol in 3GPP TS 32.295.
- ***E*** (Extension Header) — This flag indicates the presence of a meaningful value of the *Next Extension Header* field.
  - 0 = The *Next Extension Header* field is either not present, or, if present, must not be interpreted.
  - 1 = The *Next Extension Header* field is present, and must be interpreted, as described below in this section.
- ***S*** (Sequence Number) — This flag indicates the presence of a meaningful value of the *Sequence Number* field.
  - 0 = The *Sequence Number* field is either not present, or, if present, must not be interpreted.
  - 1 = The *Sequence Number* field is present, and must be interpreted, as described below in this section.
- ***PN*** (N-PDU Number) — This flag indicates the presence of a meaningful value of the *N-PDU Number* field.
  - 0 = The *N-PDU Number* field is either not present, or, if present, must not be interpreted.
  - 1 = The *N-PDU Number* field is present, and must be interpreted, as described below in this section.
- ***Message Type*** — This field indicates the type of GTP message. The E810 factory parsing program indicates if the *Message Type* field value is equal to 255 (0xFF) which means the packet is carrying user data (G-PDU).

- ***Length*** — This field indicates the length in octets of the payload. That is, the rest of the packet following the mandatory part of the GTP header (that is the first 8 octets). The *Sequence Number*, the *N-PDU Number* or any Extension headers must be considered to be part of the payload (in other words, included in the length count).
- ***Tunnel Endpoint ID*** (TEID) — This field unambiguously identifies a tunnel endpoint in the receiving GTP-U or GTP-C protocol entity. The receiving end side of a GTP tunnel locally assigns the TEID value the transmitting side has to use. The TEID values are exchanged between tunnel endpoints using GTP-C (or RANAP, over the Iu) messages.

The following fields are optional fields in GTPv1 header:

- ***Sequence Number*** — This field is an optional field in G-PDUs. It is used as a transaction identity for signaling messages having a response message defined for a request message, that is, the *Sequence Number* value is copied from the request to the response message header. In the user plane, an increasing sequence number for T-PDUs is transmitted via GTP-U tunnels, when transmission order must be preserved.
- ***N-PDU Number*** — This field is used at the Inter SGSN Routing Area Update procedure and some inter-system handover procedures (for example, between 2G and 3G radio access networks). This field is used to coordinate the data transmission for acknowledged mode of communication between the MS and the SGSN. The exact meaning of this field depends upon the scenario. For example, for GSM/GPRS to GSM/GPRS, the SNDCP N-PDU number is present in this field.
- ***Next Extension Header Type*** — This field defines the type of Extension Header that follows this field in the GTP-PDU.

A.5.5.8.2 GTPv2

GTPv2 is specified in 3GPP TS 29.274 and the information below is based on version 14.2.0 published in December, 2016.

The frame format for GTPv2 is described in Figure A-46.

Note: GTPv2 is only specified for GTP-C packets.

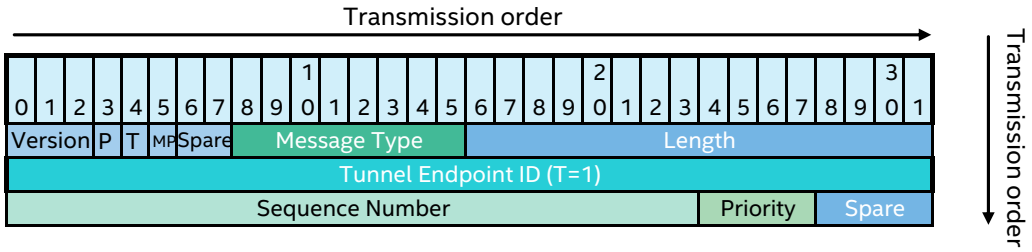


Figure A-46. GTPv2 Header Format

The following fields are always present fields in GTPv2 header:

- ***Version*** — This field is used to determine the version of the GTP protocol. The version number must be set to "2".
- ***P*** (Piggybacking) — When set, another GTPv2-C message is present at the end of this one.
- ***T*** (TEID) — Is set when the header includes a TEID field.
- ***MP*** — When this flag is set, the GTPv2-C header includes a priority field.
- ***Spare*** — Unused. Must be set to "0" by the sender and be ignored by the receiver.

- ***Message Type*** — Set to the unique value for each type of control plane message.
- ***Message Length*** — This field must indicate the length of the message in octets excluding the mandatory part of the GTP-C header (the first 4 octets). The TEID (if present) and the *Sequence Number* must be included in the length count.

  A piggybacked initial message and the preceding triggered response message present in the common IP/UDP packet must have their own length and sequence number in their respective GTP-C headers. The overall length of the IP/UDP packet must indicate the total length of the two GTP-C messages.
- ***Tunnel Endpoint ID*** (TEID) (only exists when T=1) — This field unambiguously identifies a tunnel endpoint in the receiving GTP-C entity. The *Tunnel Endpoint ID* is set by the sending entity in the GTP header of all control plane messages to the TEID value provided by the corresponding receiving entity. If a peer's TEID is not available, the TEID field must be present in a GTPv2-C header, but its value must be set to "0".

  Note: The TEID in the GTP header of a Triggered (or Triggered Reply) message is set to the TEID value provided by the corresponding receiving entity regardless of whether the source IP Address of the initial (or triggered) message and the IP Destination Address provided by the receiving entity for subsequent control plane Initial messages are the same.
- ***Sequence Number*** — Same as in GTPv1.
- ***Priority*** — Relative priority of the GTP-C message. Only relevant if the *MP* flag is set to 1). It must be encoded as the binary value of the Message Priority and it can take any value between 0 and 15, where 0 corresponds to the highest priority and 15 to the lowest priority.

  If the *MP* flag is set to "0" in Octet 1, bits 8 to 5 of octet 12 must be set to "0" by the sending entity and ignored by the receiving entity.
- ***Spare*** — Spare bits. The sending entity must set them to "0" and the receiving entity must ignore them.

NOTE: ***This page intentionally left blank.***

LEGAL